Arria 10 Transceiver PHY User Guide

Subscribe
Send Feedback

UG-A10XCVR
2013.12.02

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

Arria 10 Transceiver PHY Overview 1-1
Device Transceiver Layout.....1-2
Arria 10 GX Device Transceiver Layout.....1-3
Arria 10 GT Device Transceiver Layout.....1-7
Arria 10 GX and GT Device Package Details.....1-11
Arria 10 SX Device Transceiver Layout.....1-12
Arria 10 SX Device Package Details.....1-14
Transceiver PHY Architecture Overview.....1-15
Transceiver Bank Architecture.....1-15
PHY Layer Transceiver Components.....1-17
Transceiver Phase-Locked Loops.....1-20
Clock Generation Block (CGB).....1-21

Implementing Protocols in Arria 10 Transceivers.....2-1
Transceiver Design IP Blocks.....2-1
Transceiver Design Flow.....2-2
Select and Instantiate PHY IP.....2-2
Configure the PHY IP.....2-3
Generate PHY IP.....2-4
Select PLL IP.....2-4
Configure PLL IP.....2-5
Generate PLL IP 2-5
Reset Controller 2-6
Create Reconfiguration Logic.....2-6
Connect PHY IP to PLL IP and Reset Controller.....2-6
Connect the Transceiver Datapath to MAC IP Core or to a Data Generator / Analyzer.....2-6
Compile Design.....2-6
Verify Design Functionality.....2-7
Arria 10 Transceiver Protocols and PHY IP Support.....2-7
Using the Arria 10 Transceiver Native PHY IP.....2-12
General and Datapath Parameters 2-15
PMA Parameters.....2-17

Enhanced PCS Parameters 2-22
Standard PCS Parameters....2-29
Dynamic Reconfiguration Parameters....2-35
Enhanced PCS and PMA Ports....2-37
Standard PCS and PMA Ports....2-52
Preset Configuration Options....2-61
IP Core File Locations....2-61
Interlaken....2-62
Metaframe Format and Framing Layer Control Word....2-64
Interlaken Configuration Clocking and Bonding....2-65
How to Implement Interlaken in Arria 10 Transceivers....2-71
Design Example....2-74
Native PHY IP Parameter Settings for Interlaken....2-75
Ethernet....2-79
Gigabit Ethernet (GbE) and GbE with 1588....2-80
10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC....2-92
10GBASE-KR PHY IP with FEC Option....2-103
1G/10 Gbps Ethernet PHY IP Core....2-149
Acronyms....2-179
PCI Express....2-179
Transceiver Channel Datapath for PIPE....2-181
Supported PIPE Features....2-182
How to Connect TX PLLs for PIPE Gen1, Gen2, and Gen3 Mode....2-191
How to Implement PCI Express in Arria 10 Transceivers....2-197
Native PHY IP Parameter Settings for PCI Express....2-198
Native PHY IP Ports for PCI Express....2-204
How to Place Channels for PIPE Configurations....2-212
Design Example....2-215
CPRI....2-216
Transceiver Channel Datapath and Clocking for CPRI....2-216
Supported Features for CPRI 2-218
Word Aligner in Manual Mode for CPRI....2-219
How to Implement CPRI in Arria 10 Transceivers....2-220
Native PHY IP Parameter Settings for CPRI....2-222
Other Protocols....2-225
Using the "Basic" and "Basic with KR FEC" Configurations of Enhanced PCS....2-225
Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS....2-232

Design Considerations for Data Rates Above 17.4 Gbps Using Arria 10 GT Channels....2-250
Simulating the Transceiver Native PHY IP Core....2-256
NativeLink Simulation Flow....2-256
Custom Simulation Flow....2-260

PLLs and Clock Networks....3-1
PLLs....3-3
ATX PLL....3-3
fPLL....3-11
CMU PLL....3-19
Input Reference Clock Sources....3-24
Dedicated Reference Clock Pins....3-25
Receiver Input Pins....3-26
PLL Cascading as a Input Reference Clock Source....3-26
Reference Clock Network....3-27
Transmitter Clock Network....3-27
x1 Clock Lines....3-27
x6 Clock Lines....3-28
xN Clock Lines....3-29
GT Clock Lines....3-31
Clock Generation Block....3-32
FPGA Fabric-Transceiver Interface Clocking....3-34
Transmitter Data Path Interface Clocking....3-34
Receiver Data Path Interface Clocking....3-36
Channel Bonding....3-38
PLL Feedback and Cascading Clock Network....3-38
Using PLLs and Clock Networks....3-40
Non-bonded configurations....3-40
Bonded configurations....3-44
PLL cascading....3-48
Mix and Match Example....3-49

Resetting Transceiver Channels....4-1
When Is Reset Required? 4-2
How Do I Reset?....4-2
Recommended Reset Sequence....4-3
Transceiver Blocks Affected by Reset and Powerdown Signals....4-8

Reset Signals for PLL, PMA, and PCS Blocks....4-8
Using the Altera Transceiver PHY Reset Controller....4-8
Parameterizing the Transceiver PHY Reset Controller IP....4-10
Transceiver PHY Reset Controller Parameters....4-10
Transceiver PHY Reset Controller Interfaces....4-12
Transceiver PHY Reset Controller Resource Utilization....4-16
Using a User-Coded Reset Controller....4-16
User-Coded Reset Controller Signals....4-16
Combining Status or PLL Lock Signals....4-18
Timing Constraints for Bonded PCS and PMA Channels....4-18

Arria 10 Transceiver PHY Architecture....5-1
Arria 10 PMA Architecture....5-1
Transmitter....5-1
Receiver....5-4
Loopback....5-13
Arria 10 Enhanced PCS Architecture....5-14
Transmitter Datapath....5-15
Receiver Datapath....5-23
Arria 10 Standard PCS Architecture....5-31
Transmitter Datapath....5-31
Receiver Datapath....5-37
Arria 10 PCI Express Gen3 PCS Architecture....5-45
Transmitter Datapath....5-47
Receiver Datapath....5-48
PIPE Interface....5-49

Reconfiguration Interface and Dynamic Reconfiguration....6-1
Ports and Parameters....6-1
Interacting with the Reconfiguration Interface....6-4
Performing a Read to the Reconfiguration Interface....6-4
Performing a Write to the Reconfiguration Interface....6-4
Reconfiguring Channel and PLL Blocks....6-5
Step 1: Generate Required Configuration Files....6-5
Step 2: Determine Address Offsets and Differences....6-7
Step 3: Perform Read-Modify-Writes....6-7
Step 4: Reset Transceiver Channels....6-7
Using Configuration Files....6-7

Transmitter PLL Switching....6-9
Switching Reference Clocks....6-10
ATX Reference Clock Switching....6-10
fPLL Reference Clock Switching....6-11
CDR and CMU Reference Clock Switching....6-12
Changing PMA Analog Parameters....6-13
Using Data Pattern Generators and Checkers....6-13
Using PRBS and Square Wave Data Pattern Generator and Checker....6-14
Unsupported Features....6-20
Transceiver and PLL Address Map....6-21

Document Revision History....7-1

Arria 10 Transceiver PHY Overview

1

2013.12.02

UG-A10XCVR | Subscribe | Send Feedback

This user guide provides details on the Arria® 10 transceiver physical (PHY) layer architecture, PLLs, clock networks and transceiver PHY IP. It also describes available transceiver features like reset controller, dynamic reconfiguration and provides protocol specific implementation details.

Altera's Arria 10 devices offer up to 96 transceivers with integrated advanced high speed analog signal conditioning and clock data recovery techniques for chip-to-chip, chip-to-module, and backplane applications.

The Arria 10 GX and SX devices have GX transceiver channels that can support data rates up to 17.4 Gbps for chip-to-chip applications and 16.0 Gbps for backplane applications

The Arria 10 GT devices have up to 16 GT transceiver channels, that can support data rates up to 28.1 Gbps for short reach chip-to-chip and chip-to-module applications. Additionally, the GT devices have GX transceiver channels that can support data rates up to 17.4 Gbps for both chip-to-chip and backplane applications. If all 16 GT channels are used, then the largest GT devices can have up to 72 GX transceiver channels.

The Arria 10 transceivers support reduced power modes with data rates up to 11.3 Gbps (chip-to-chip) and 10.3125 Gbps (backplane) for critical power sensitive designs. In GX devices that have transceivers on both sides of the device, each side can be operated independently in standard and reduced power modes.

Table 1-1: Data Rates Supported by GX Transceiver Channel Type

Device Variant	Standard Power Mode(1)		Reduced Power Mode(1)	
	Chip-to-Chip	Backplane	Chip-to-Chip	Backplane
SX(2)	611 Mbps to 17.4 Gbps	611 Mbps to 16.0 Gbps	611 Mbps to 11.3 Gbps	611 Mbps to 10.3125 Gbps
GX(2)	611 Mbps to 17.4 Gbps	611 Mbps to 16.0 Gbps	611 Mbps to 11.3 Gbps	611 Mbps to 10.3125 Gbps
GT(3)	611 Mbps to 17.4 Gbps	611 Mbps to 17.4 Gbps	611 Mbps to 11.3 Gbps	611 Mbps to 10.3125 Gbps

(1) To operate GX transceiver channels at designated data rates in standard and reduced power modes, apply the corresponding core and periphery power supplies. Refer to Arria 10 Device Datasheet for more details.
(2) For SX and GX device variants, the maximum transceiver data rates are specified for the fastest (-1) transceiver speed grade.
(3) For GT device variants, the maximum transceiver data rates are specified for (-2) transceiver speed grade.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Table 1-2: Data Rates Supported by GT Transceiver Channel Type

Device Variant	Data Rates[4]	
	Chip-to-Chip	Backplane
GT[3]	611 Mbps to 28.1 Gbps	611 Mbps to 17.4 Gbps

Note: The device data rates are dependent on the device speed grade. Refer to *Arria 10 Device Datasheet* for details on available speed grades and supported data rates.

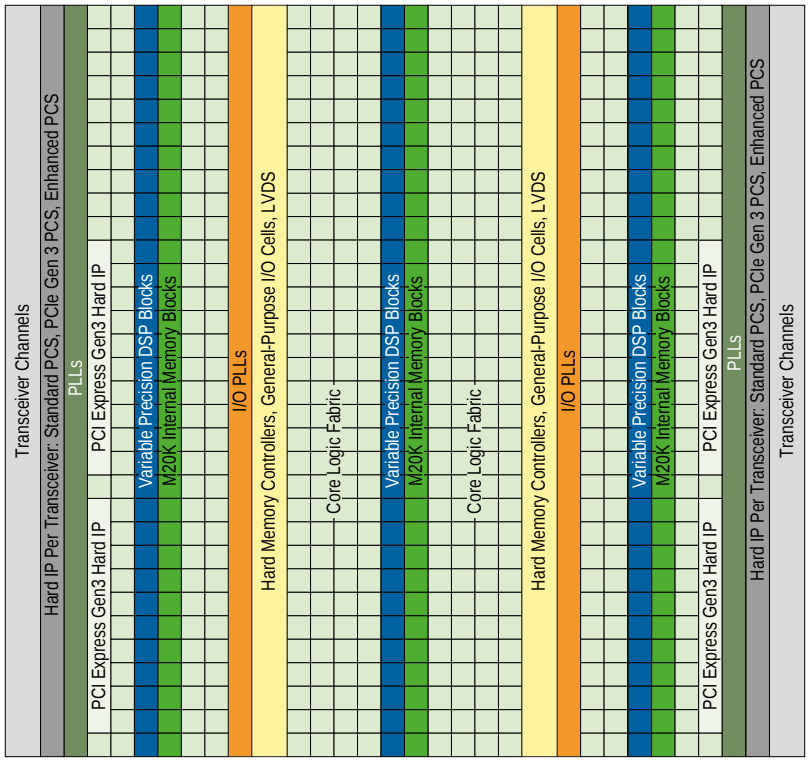
Related Information

Arria 10 Device Datasheet

Device Transceiver Layout

Figure 1-1: Arria 10 FPGA Architecture Block Diagram

The transceiver channels are placed on the left side periphery in most Arria 10 devices. For larger Arria 10 devices, additional transceiver channels are placed on the right side periphery.


Transceiver Channels
Hard IP Per Transceiver: Standard PCS, PCIe Gen 3 PCS, Enhanced PCS
PLLs
PCI Express Gen3 Hard IP
PCI Express Gen3 Hard IP
Variable Precision DSP Blocks
M20K Internal Memory Blocks
I/O PLLs
Hard Memory Controllers, General-Purpose I/O Cells, LVDS
Core Logic Fabric
Variable Precision DSP Blocks
M20K Internal Memory Blocks
Core Logic Fabric
Hard Memory Controllers, General-Purpose I/O Cells, LVDS
I/O PLLs
Variable Precision DSP Blocks
M20K Internal Memory Blocks
PCI Express Gen3 Hard IP
PCI Express Gen3 Hard IP
PLLs
Hard IP Per Transceiver: Standard PCS, PCIe Gen 3 PCS, Enhanced PCS
Transceiver Channels

[4] Because the GT transceiver channels are designed for peak performance, they do not have a reduced power mode of operation.

Arria 10 GX Device Transceiver Layout

The largest Arria 10 GX device includes 96 transceiver channels. A column array of eight transceiver banks on the left and the right side periphery of the device is shown in the following figure. Each transceiver bank has six transceiver channels. Some devices have transceiver banks with only three channels. The transceiver banks with only three channels are the uppermost transceiver banks. Arria 10 devices also include PCI Express Hard IP blocks.

The figures below illustrate different transceiver bank layouts for Arria 10 GX device variants.

Figure 1-2: Arria 10 GX Devices with 96 Transceiver Channels and Four PCIe Hard IP Blocks

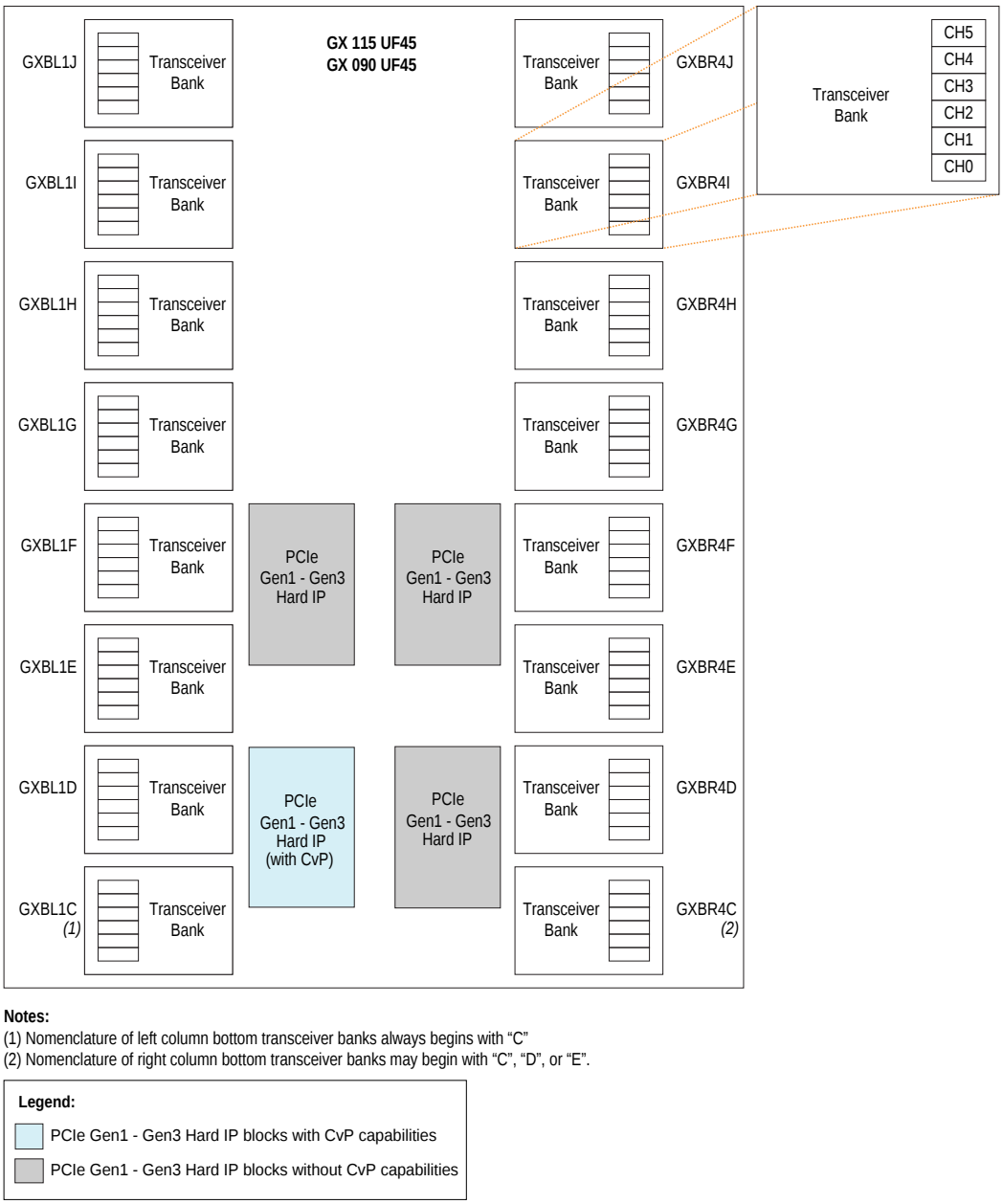
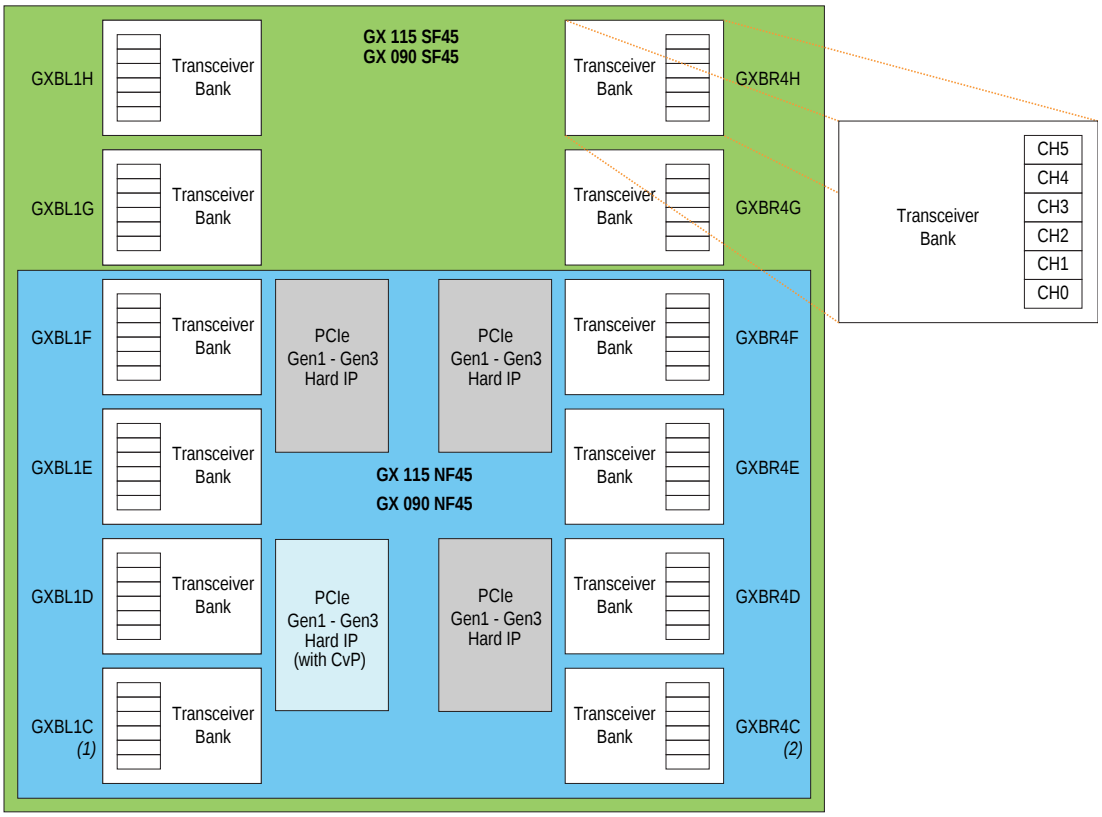


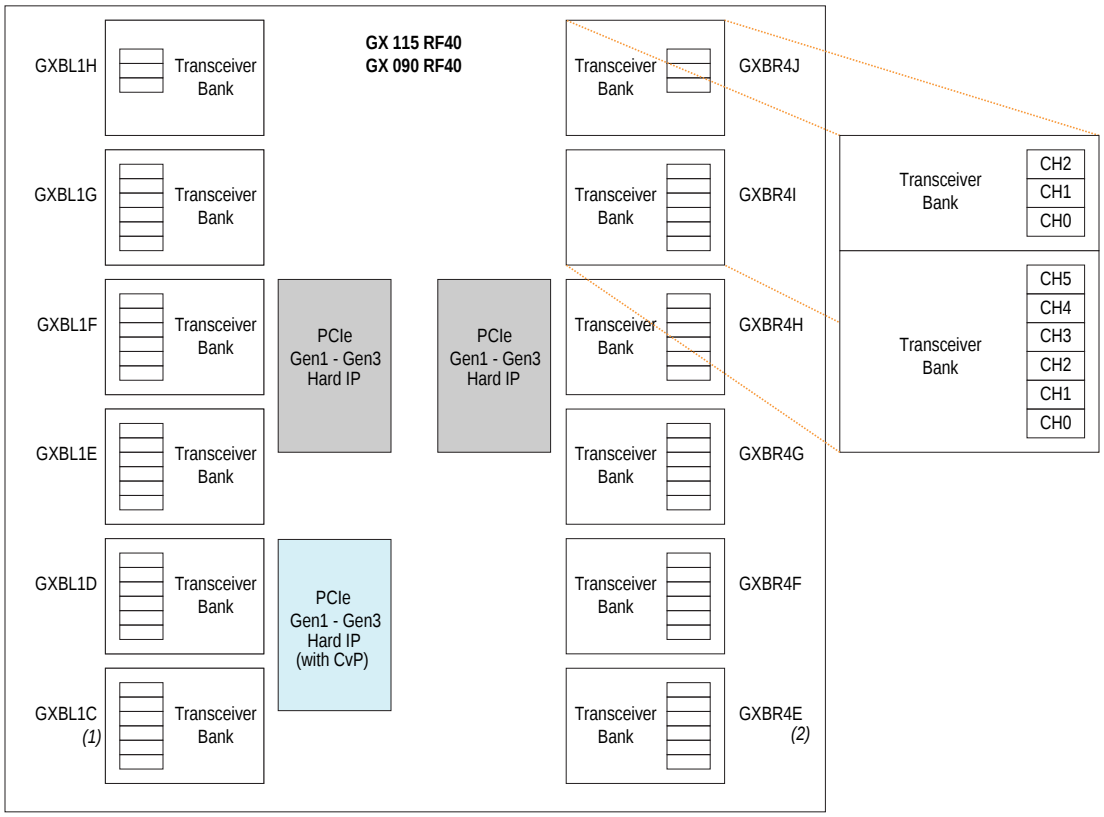
Figure 1-3: Arria 10 GX Devices with 72 and 48 Transceiver Channels and Four PCIe Hard IP Blocks.



Notes:
(1) Nomenclature of left column bottom transceiver banks always begins with "C"
(2) Nomenclature of right column bottom transceiver banks may begin with "C", "D", or "E".

Legend:
- PCIe Gen1 - Gen3 Hard IP blocks with CvP capabilities
- PCIe Gen1 - Gen3 Hard IP blocks without CvP capabilities
- Arria 10 GX device with 48 transceiver channels and four PCIe Hard IP blocks.
- Arria 10 GX device with 72 transceiver channels and four PCIe Hard IP blocks.

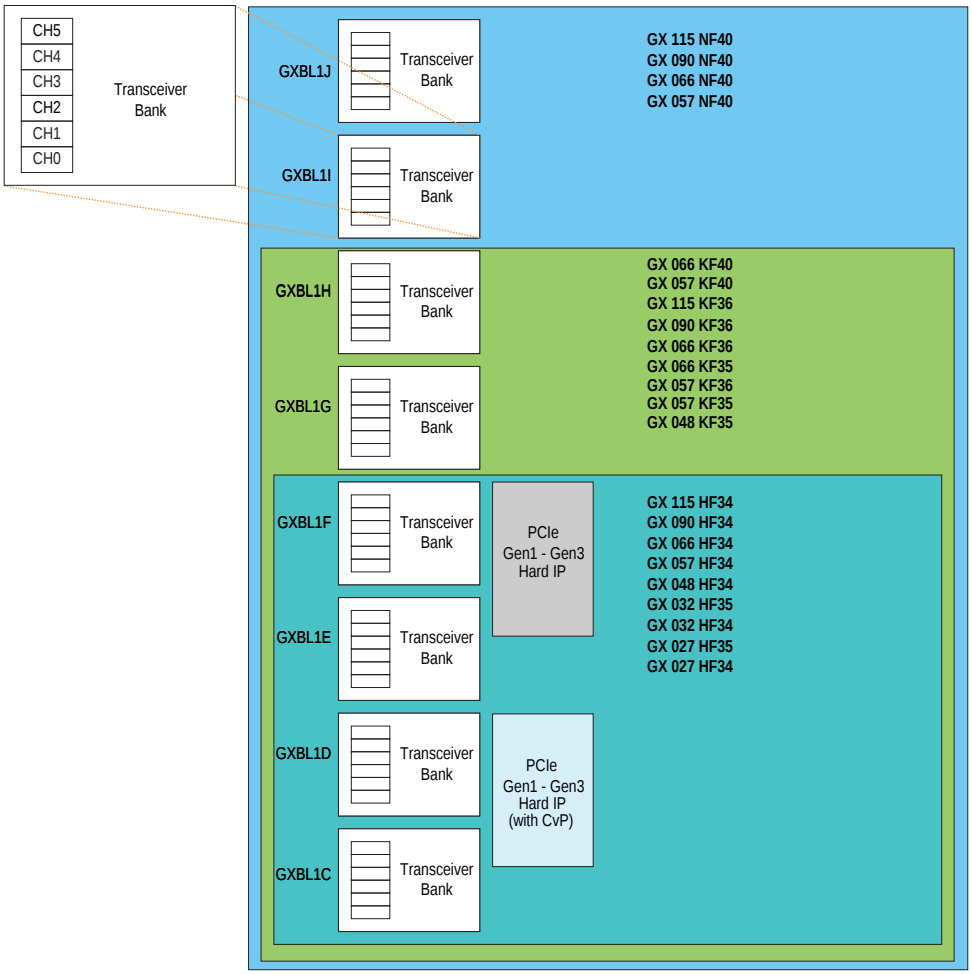
Figure 1-4: Arria 10 GX Devices with 66 Transceiver Channels and Three PCIe Hard IP Blocks



Notes:
(1) Nomenclature of left column bottom transceiver banks always begins with "C"
(2) Nomenclature of right column bottom transceiver banks may begin with "C", "D", or "E".

Legend:
- PCIe Gen1 - Gen3 Hard IP blocks with CvP capabilities
- PCIe Gen1 - Gen3 Hard IP blocks without CvP capabilities

Figure 1-5: Arria 10 GX Devices with 48, 36, and 24 Transceiver Channels and Two PCIe Hard IP Blocks



Note:
(1) These devices have transceivers only on the left hand side of the device.

Legend:
- PCIe Gen1 - Gen3 Hard IP blocks with CvP capabilities
- PCIe Gen1 - Gen3 Hard IP blocks without CvP capabilities
- Arria 10 GX device with 48 transceiver channels and two PCIe Hard IP blocks.
- Arria 10 GX device with 36 transceiver channels and two PCIe Hard IP blocks.
- Arria 10 GX device with 24 transceiver channels and two PCIe Hard IP blocks.

Figure 1-6: Arria 10 GX Devices with 12 Transceiver Channels and One PCIe Hard IP Block

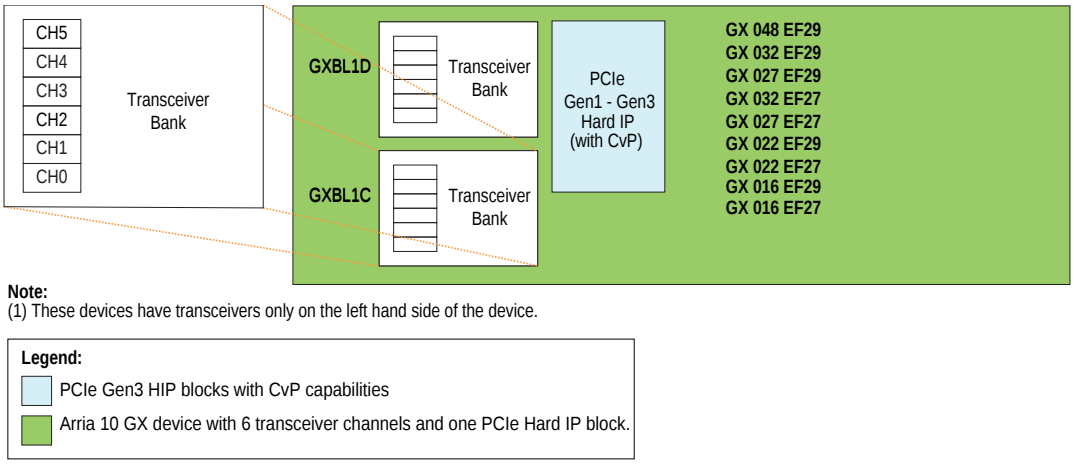
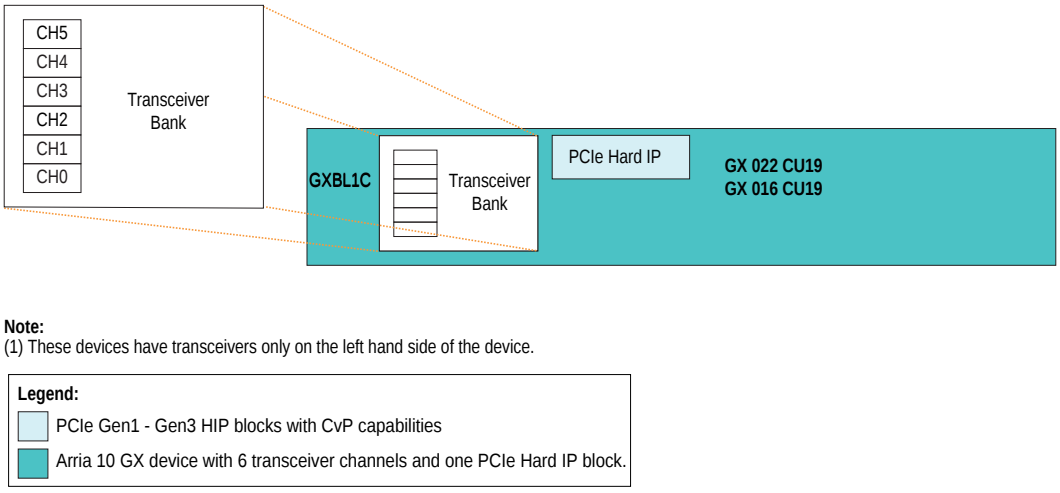


Figure 1-7: Arria 10 GX Devices with 6 Transceiver Channels and One PCIe Hard IP Block

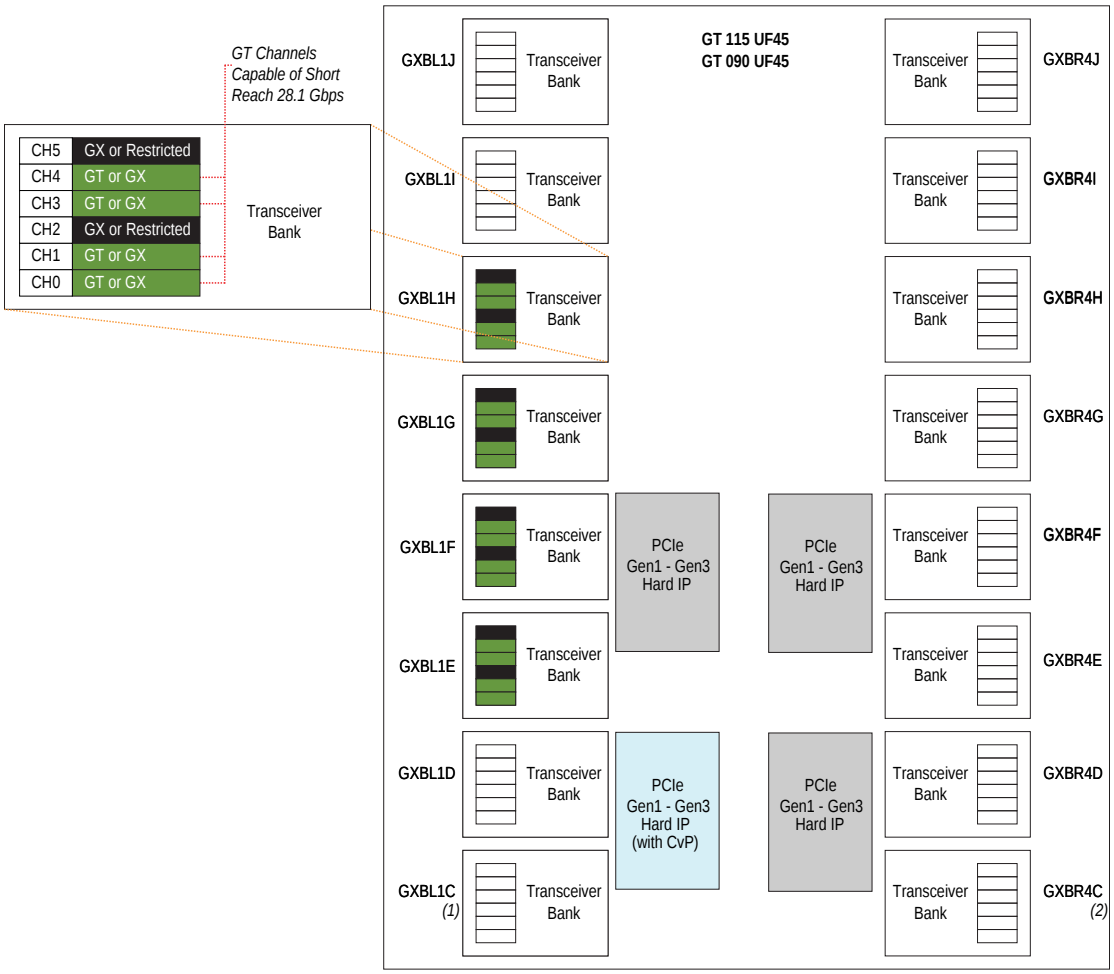


Arria 10 GT Device Transceiver Layout

The largest GT device has 96 transceiver channels and four PCI Express Hard IP blocks. All GT devices have a total of 16 GT transceiver channels that can support data rates up to 28.1 Gbps.

In GT devices, transceiver banks GXBL1E, GXBL1F, GXBL1G, and GXBL1H each contain four GT transceiver channels. These are channels 0, 1, 3 and 4. The channels 2 and 5 are GX transceiver channels.

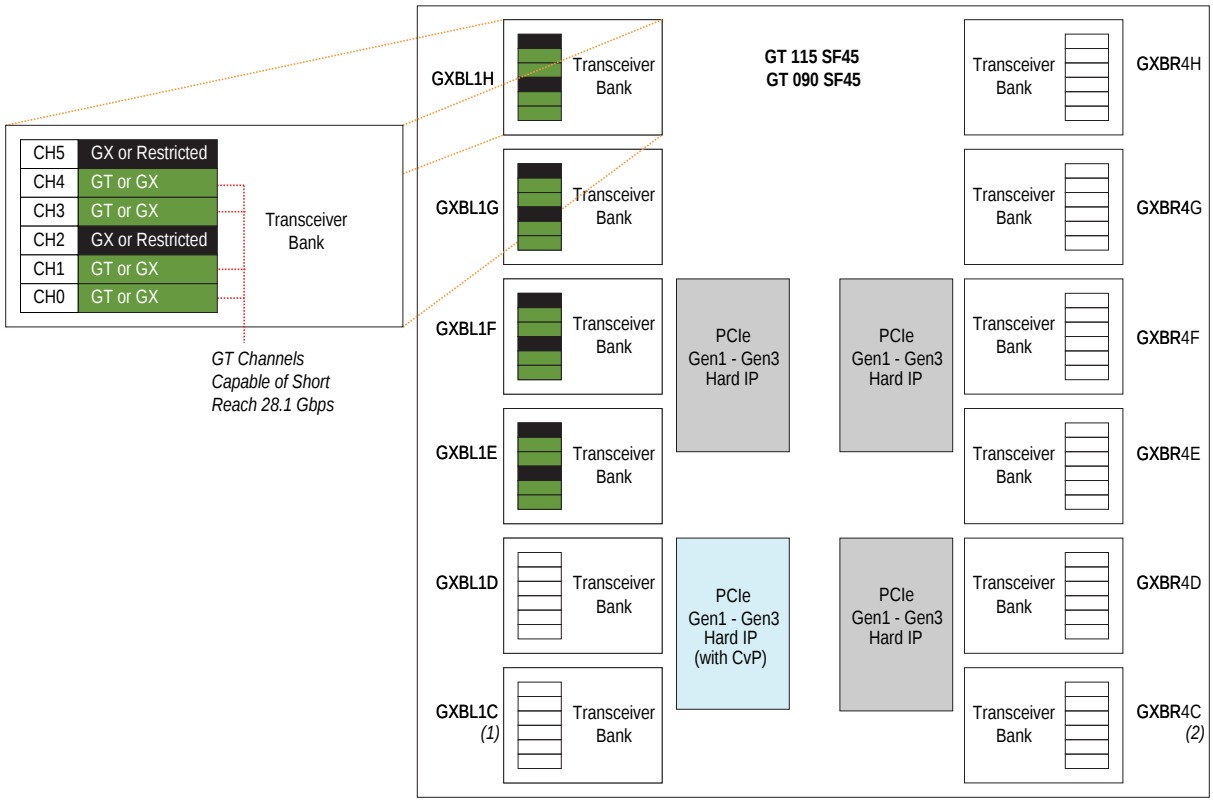
Figure 1-8: Arria 10 GT Devices with 96 Transceiver Channels and Four PCIe Hard IP Blocks



Notes:
(1) Nomenclature of left column bottom transceiver banks always begins with "C"
(2) Nomenclature of right column bottom transceiver banks may begin with "C", "D", or "E".

Legend:
- GT transceiver channels (channel 0, 1, 3, and 4)
- Transceiver channels that cannot be used (channel 2 and 5) when all four GT channels are used.
- PCIe Gen1 - Gen3 Hard IP blocks with CvP capabilities
- PCIe Gen1 - Gen3 Hard IP blocks without CvP capabilities

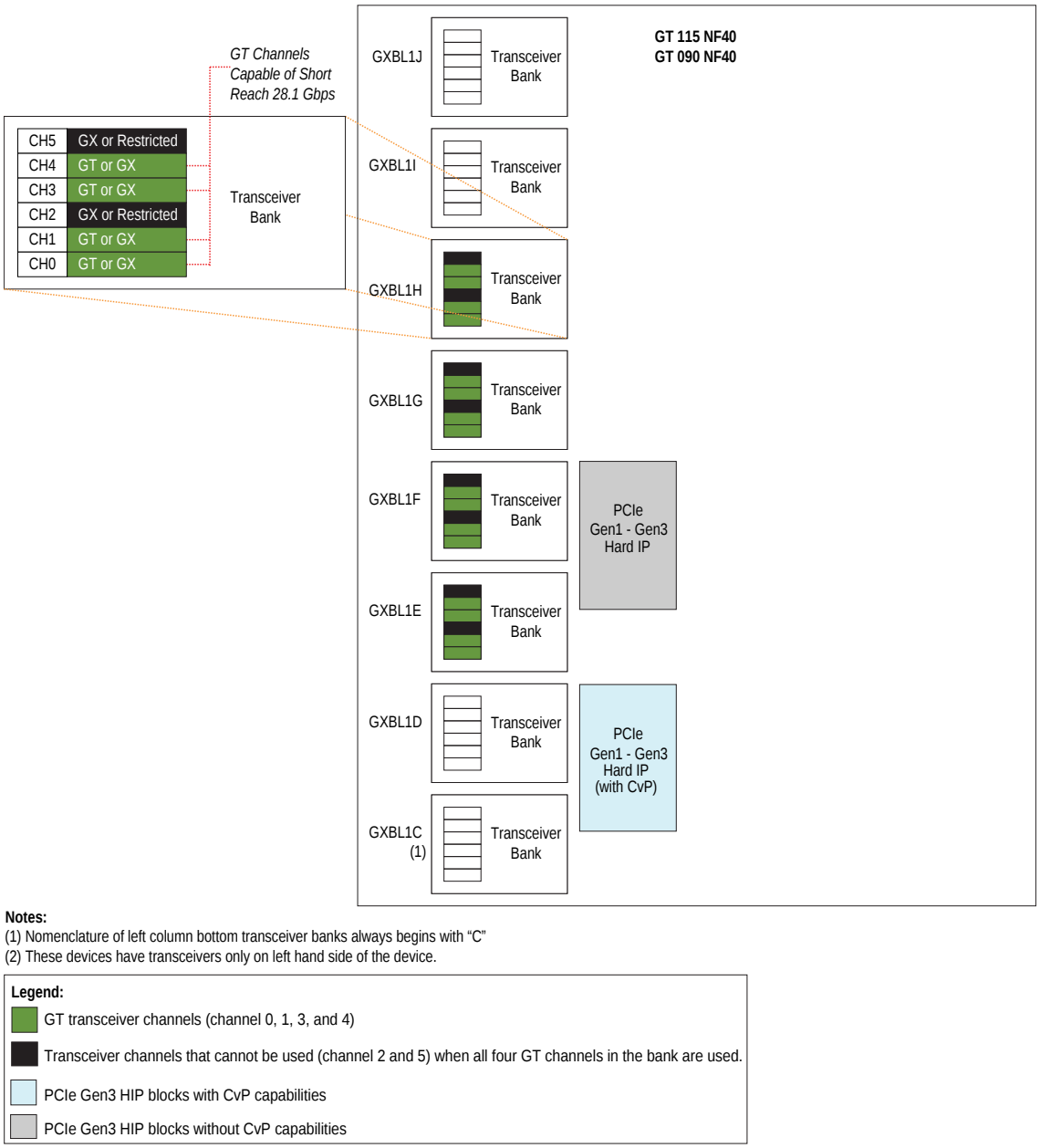
Figure 1-9: Arria 10 GT Devices with 72 Transceiver Channels and Four PCIe Hard IP Blocks



Notes:
(1) Nomenclature of left column bottom transceiver banks always begins with "C"
(2) Nomenclature of right column bottom transceiver banks may begin with "C", "D", or "E".

Legend:
- GT transceiver channels (channel 0, 1, 3, and 4)
- Transceiver channels that cannot be used (channel 2 and 5) when all four GT channels in the bank are used.
- PCIe Gen1 - Gen3 Hard IP blocks with CvP capabilities
- PCIe Gen1 - Gen3 Hard IP blocks without CvP capabilities

Figure 1-10: Arria 10 GT Devices with 48 Transceiver Channels and Two PCIe Hard IP Blocks


GT 115 NF40
GT 090 NF40
GT Channels Capable of Short Reach 28.1 Gbps
CH5 GX or Restricted
CH4 GT or GX
CH3 GT or GX
CH2 GX or Restricted
CH1 GT or GX
CH0 GT or GX
Transceiver Bank
GXBL1J
GXBL1I
GXBL1H
GXBL1G
GXBL1F
GXBL1E
GXBL1D
GXBL1C (1)
PCIe Gen1 - Gen3 Hard IP
PCIe Gen1 - Gen3 Hard IP (with CvP)
Notes:
(1) Nomenclature of left column bottom transceiver banks always begins with "C"
(2) These devices have transceivers only on left hand side of the device.
Legend:
GT transceiver channels (channel 0, 1, 3, and 4)
Transceiver channels that cannot be used (channel 2 and 5) when all four GT channels in the bank are used.
PCIe Gen3 HIP blocks with CvP capabilities
PCIe Gen3 HIP blocks without CvP capabilities

The largest GT device has 96 transceiver channels, which include 16 GT transceiver channels supporting data rates greater than 17.4 Gbps. If all 16 GT transceiver channels are used, then there will be 72 GX transceiver channels that can drive backplanes at data rates up to 17.4 Gbps and 8 GX channels that are unusable. In contrast, the GX transceiver channels in SX and GX device variants can drive backplanes at data rates up to 16.0 Gbps.

In GT devices that have transceivers on both sides of the device, the GX transceiver channels on right side can be used in reduced power mode. In GT devices, where none of the GT channels are used, the transceiver channels can be used as GX channels in standard or reduced power mode.

Related Information

Arria 10 GT Channel Usage on page 2-251
For details about Arria 10 GT channel usage guidelines

Arria 10 GX and GT Device Package Details

The following tables list package sizes, available transceiver channels, and PCI Express Hard IP blocks for Arria 10 GX and GT devices.

Table 1-3: Package Details for GX and GT Devices with Transceivers and HIP Blocks Located on the Left Side Periphery of the Device.

Device	U19[5]	F27 [6]	F29 [7]	F34 [8]	F35 [8]	F36 [8]	F40 [9]	F40 [9]
	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count
GX 016	6, 1	12, 1	12, 1					
GX 022	6, 1	12, 1	12, 1					
GX 027		12, 1	12, 1	24, 2	24, 2			
GX 032		12, 1	12, 1	24, 2	24, 2			
GX 048			12, 1	24, 2	36, 2			
GX 057				24, 2	36, 2	36, 2	36, 2	48, 2
GX 066				24, 2	36, 2	36, 2	36, 2	48, 2
GX 090				24, 2		36, 2		48, 2
GX 115				24, 2		36, 2		48, 2
GT 090								48, 2
GT 115								48, 2

Table 1-4: Package Details for GX and GT Devices with Transceivers and Hard IP Blocks Located on the Left and Right Side Periphery of the Device.

Device	F40 [9]	F45 [10]	F45 [10]	F45 [10]
	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count
GX 090	66, 3	48, 4	72, 4	96, 4

[5] Package U19: 19mm x 19mm package; 484 pins.
[6] Package F27: 27mm x 27mm package; 672 pins.
[7] Package F29: 29mm x 29mm package; 780 pins.
[8] Packages F34, F35, and F36: 35 mm x 35 mm package size; 1152 pins.
[9] Package F40: 40 mm x 40 mm package size; 1517 pins.

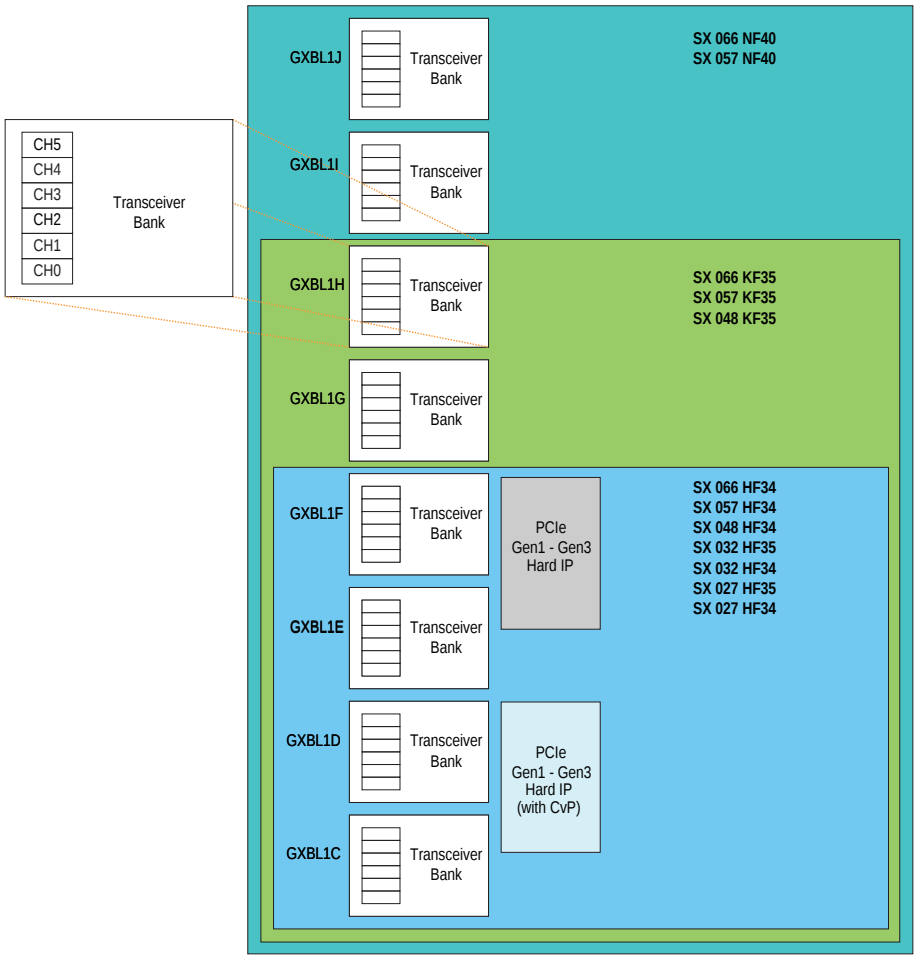
Device	F40 (9)	F45 (10)	F45 (10)	F45 (10)
	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count
GX 115	66, 3	48, 4	72, 4	96, 4
GT 090			72, 4	96, 4
GT 115			72, 4	96, 4

Arria 10 SX Device Transceiver Layout

The largest SX device includes 48 transceiver channels. All SX devices have GX transceiver channel type. The transceiver banks in SX devices are located on the left side periphery of the device.

(10) Package F45: 45mm x 45mm package size; 1932 pins.

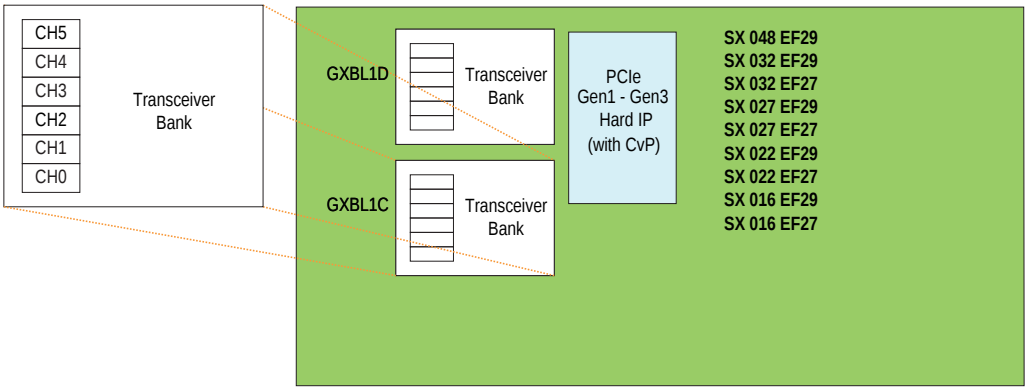
Figure 1-11: Arria 10 SX Device with 48, 36, and 24 Transceiver Channels and Two Hard IP Blocks



Note:
(1) These devices have transceivers only on the left hand side of the device.

Legend:
- PCIe Gen 3 HIP blocks with CvP capabilities
- PCIe Gen 3 HIP blocks without CvP capabilities
- Arria 10 SX device with 24 transceiver channels and two PCIe Hard IP blocks.
- Arria 10 SX device with 36 transceiver channels and two PCIe Hard IP blocks.
- Arria 10 SX device with 48 transceiver channels and two PCIe Hard IP blocks.

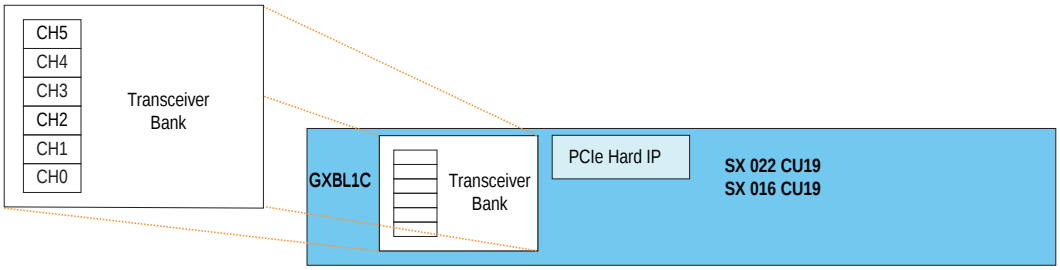
Figure 1-12: Arria 10 SX Device with 12 and 6 Transceiver Channels and One Hard IP Block



Note:
(1) These devices have transceivers only on the left hand side of the device.

Legend:
- PCIe Gen3 HIP blocks with CvP capabilities
- Arria 10 SX device with 12 transceiver channels and one Hard IP block

Figure 1-13: Arria 10 SX Device with Six Transceiver Channels and One Hard IP Block



Note:
(1) These devices have transceivers only on the left side of the device.

Legend:
- PCIe Gen1 - Gen3 HIP blocks with CvP capabilities
- Arria 10 SX device with 6 transceiver channels and 1 PCIe Hard IP block

Arria 10 SX Device Package Details

The following tables list package sizes, available transceiver channels, and PCI Express Hard IP blocks for Arria 10 SX devices.

Table 1-5: Package Details for SX Devices with Transceivers and HIP Blocks Located on the Left Side Periphery of the Device

Device	U19 (11)	F27 (12)	F29 (13)	F34(14)	F35(14)	F40(15)	F40(15)
	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count	Transceiver Count, PCIe Hard IP Block Count
SX 016	6, 1	12, 1	12, 1				
SX 022	6, 1	12, 1	12, 1				
SX 027		12, 1	12, 1	24, 2	24, 2		
SX 032		12, 1	12, 1	24, 2	24, 2		
SX 048			12, 1	24, 2	36, 2		
SX 057				24, 2	36, 2	36, 2	48, 2
SX 066				24, 2	36, 2	36, 2	48, 2

Transceiver PHY Architecture Overview

A link is defined as a single entity communication port. A link can have one or more transceiver channels. A transceiver channel is synonymous with a transceiver lane.

For example, a 10GBASE-R link has one transceiver channel or lane with a data rate of 10.3125 Gbps. A 40GBASE-R link has four transceiver channels. Each transceiver channel operates at a lane data rate of 10.3125 Gbps. Four transceiver channels give a total collective link bandwidth of 41.25 Gbps (40 Gbps before and after 64B/66B PCS encoding and decoding).

Transceiver Bank Architecture

The transceiver bank is the fundamental unit that contains all the functional blocks related to the device's high speed serial transceivers.

Each transceiver bank includes six transceiver channels in all devices except for the devices with 66 transceiver channels. These devices (with 66 transceiver channels) have both six channel and three channel transceiver banks. The uppermost transceiver bank on the left and the right side of these devices is a three channel transceiver bank. All other devices contain six channel transceiver banks.

The figures below show the transceiver bank architecture with the phase locked loop (PLL) and clock generation block (CGB) resources available in each bank.

(11) Package U19: 19mm x 19mm package; 484 pins.
(12) Package F27: 27mm x 27mm package; 672 pins.
(13) Package F29: 29mm x 29mm package; 780 pins.
(14) Packages F34 and F35: 35 mm x 35 mm package size ; 1152 pins.
(15) Package F40: 40 mm x 40 mm package size ; 1517 pins.

Figure 1-14: Three-Channel GX Transceiver Bank Architecture

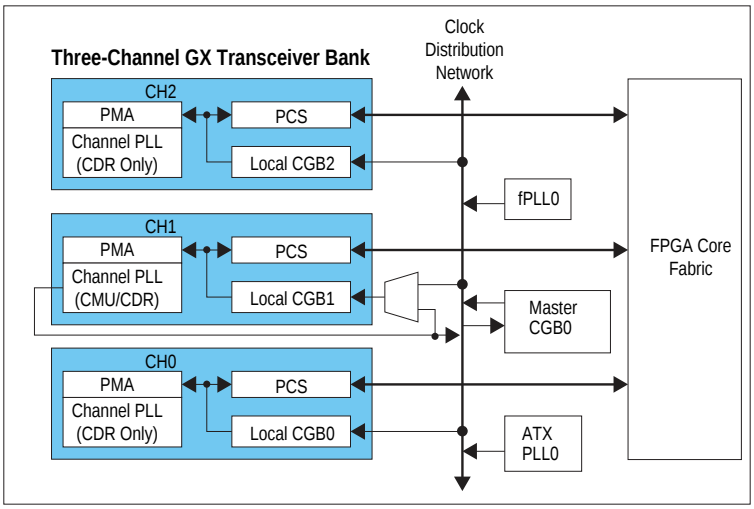


Figure 1-15: Six-Channel GX Transceiver Bank Architecture

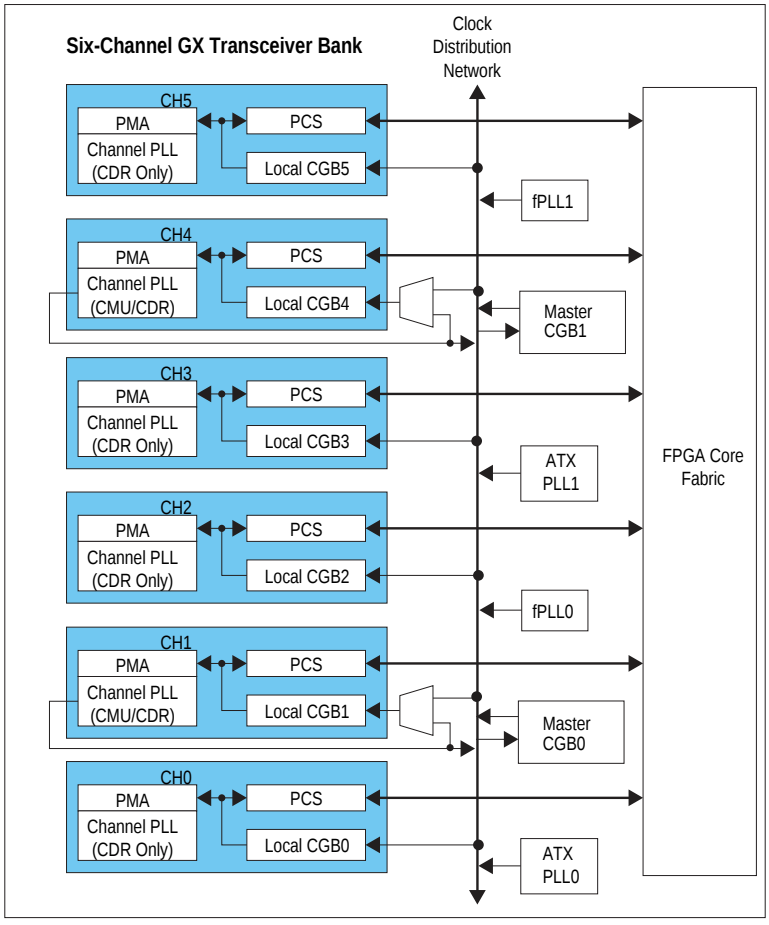
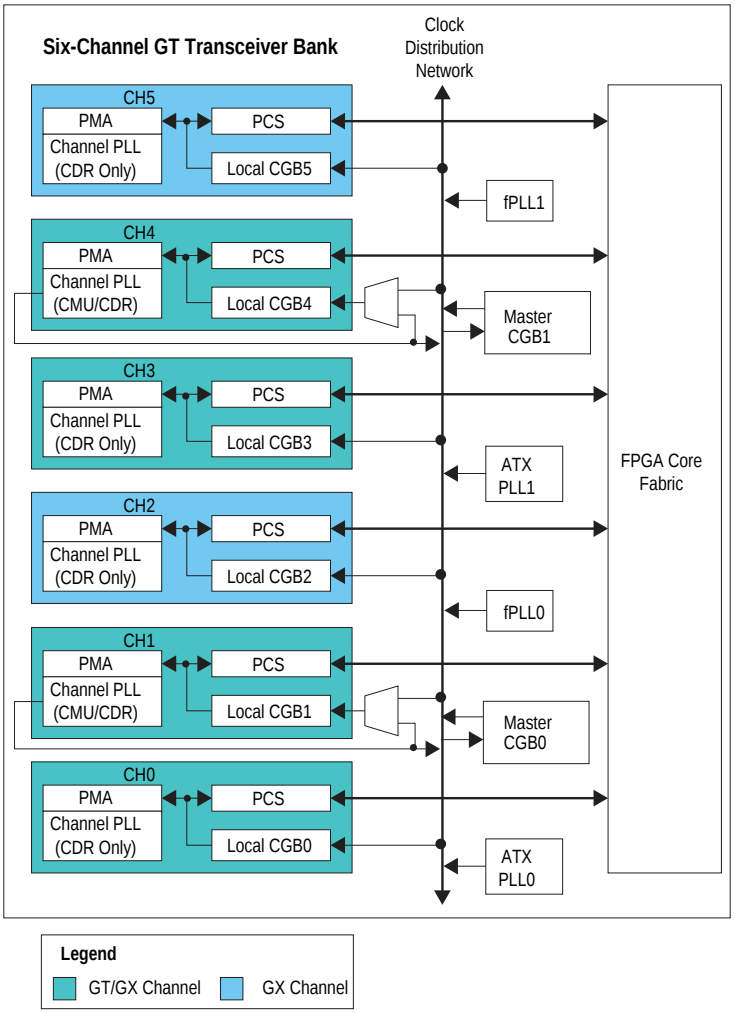


Figure 1-16: GT Transceiver Bank Architecture

In GT devices, the transceiver banks GXBL1E, GXBL1F, GXBL1G, and GXBL1H include GT channels.



The transceiver channels perform all the required PHY layer functions between the FPGA fabric and the physical medium. The high speed clock required by the transceiver channels is generated by the transceiver PLLs. The master and local clock generation blocks (CGBs) provide the necessary high speed serial and low speed parallel clocks to drive the non-bonded and bonded channels in the transceiver bank.

Related Information

Transceiver Basics
Online training course for transceivers.

PHY Layer Transceiver Components

Transceivers in Arria 10 devices support both Physical Medium Attachment (PMA) and Physical Coding Sublayer (PCS) functions at the physical (PHY) layer.

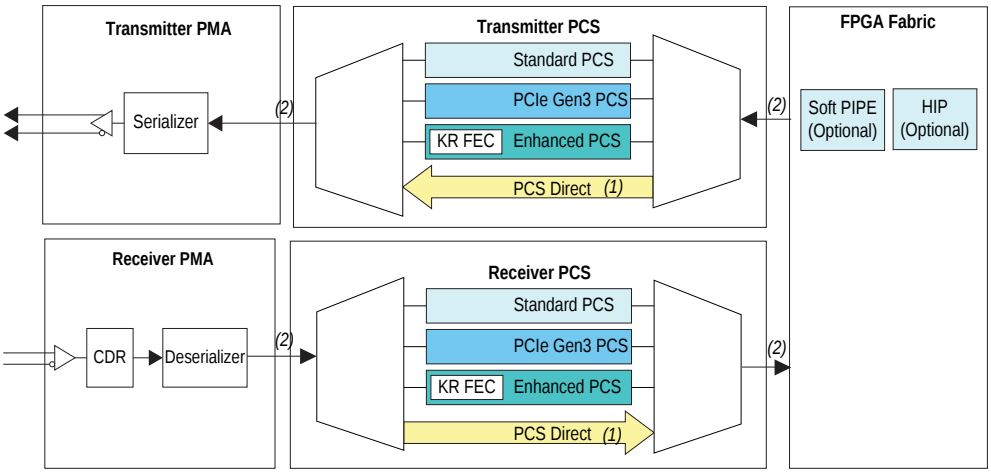
A PMA is the transceiver's electrical interface to the physical medium. The transceiver PMA consists of standard blocks such as:

- serializer/deserializer (SERDES)
- clock and data recovery PLL
- analog front end transmit drivers
- analog front end receive buffers

The PCS can be bypassed with a PCS-Direct configuration. Both the PMA and PCS blocks are fed by multiple clock networks driven by high performance PLLs. In PCS-Direct configuration, the data flow is through the PCS block, but all the internal PCS blocks are bypassed. In this mode, the PCS functionality is implemented in the FPGA fabric.

The GX Transceiver Channel

Figure 1-17: GX Transceiver Channel in Full Duplex Mode.



Notes:
(1) PCS Direct support will be available in a future release of the Quartus-II software.
(2) The FPGA Fabric - PCS and PCS-PMA interface widths are configurable.

Arria 10 GX transceiver channels have three types of PCS blocks that together support continuous data rates between of 611 Mbps to 17.4 Gbps

Table 1-6: PCS Types Supported by GX Transceiver Channels

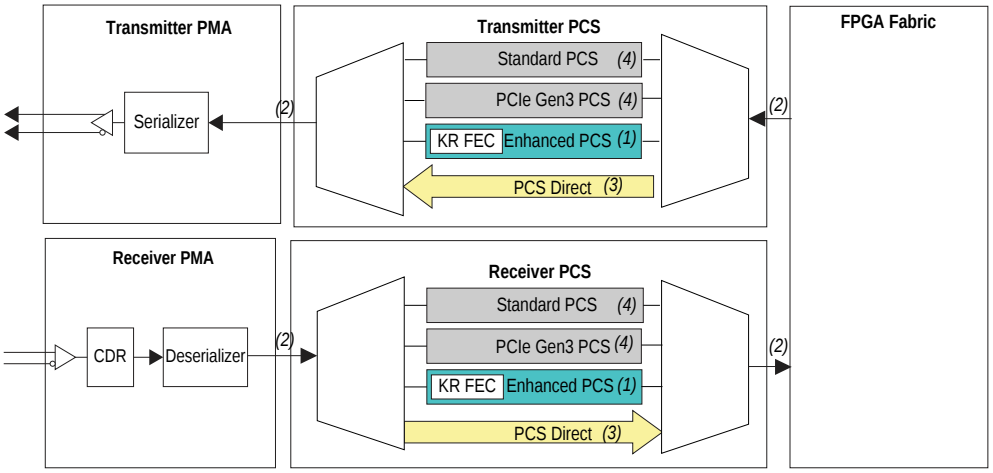
PCS Type	Data Rate
Standard PCS	611 Mbps up to 10 Gbps
Enhanced PCS	611 Mbps up to 17.4 Gbps
PCIe Gen3 PCS	8 Gbps

Note: The GX channel can also operate in PCS Direct configuration for data rates from 611 Mbps to 17.4 Gbps.

The GT Transceiver Channel

The GT transceiver channels are used for supporting data rates from 17.4 Gbps to 28.1 Gbps. The GT transceiver channels can also be reconfigured as GX transceiver channels. When they are reconfigured as GX transceiver channels, the Standard PCS, Enhanced PCS, and PCIe Gen3 PCS are available and they support data rates from 611 Mbps to 17.4 Gbps.

Figure 1-18: GT Transceiver Channel in Full Duplex Mode Operating Between 17.4 Gbps and 28.1 Gbps



Notes:
(1) The Enhanced PCS must be configured in lowl latency mode to support data rate range from 17.4 Gbps to 28.1 Gbps.
(2) The FPGA Fabric - PCS and PCS-PMA interface widths are configurable.
(3) PCS Direct support will be available in a future release of the Quartus-II software.
(4) The Standard PCS and PCIe Gen3 PCS blocks are available when the GT channel is reconfigured as a GX transceiver channel.

Table 1-7: PCS Types and Data Rates Supported by GT Channel Configurations

GT Channel Configuration	PCS Type	Data Rates Supported
GT	Standard PCS	Not available
	Enhanced PCS	17.4 Gbps to 28.1 Gbps[16]
	PCIe Gen3 PCS	Not available
GX	Standard PCS	611 Mbps to 10 Gbps
	Enhanced PCS	611 Mbps to 17.4 Gbps
	PCIe Gen3 PCS	8 Gbps

Note: The GT channels can also operate in PCS-Direct configuration for data rates between 611 Mbps to 28.1 Gbps.

[16] The Enhanced PCS must be configured in low latency mode to support data rate range from 17.4 Gbps to 28.1 Gbps.

Transceiver Phase-Locked Loops

Each transceiver channel in Arria 10 devices has direct access to three types of high performance PLLs:

- Advanced Transmit (ATX) PLL
- Fractional PLL (fPLL)
- Channel PLL / Clock Multiplier Unit (CMU) PLL.

These transceiver PLLs along with the Master or Local Clock Generation Blocks (CGB) drive the transceiver channels.

Related Information

PLLs on page 3-3
For more information on transceiver PLLs in Arria 10 devices.

Advanced Transmit (ATX) PLL

An advanced transmit (ATX) PLL is a high performance PLL. It supports both integer frequency synthesis and coarse resolution fractional frequency synthesis. The ATX PLL is the transceiver channel's primary transmit PLL. It can operate over the full range of supported data rates required for high data rate applications.

Related Information

- **ATX PLL** on page 3-3
  For more information on ATX PLL.
- **ATX PLL IP** on page 3-6
  For details on implementing the ATX PLL IP.

Fractional PLL (fPLL)

A fractional PLL (fPLL) is an alternate transmit PLL used for generating low clock frequencies for low data rate applications. fPLLs support both integer frequency synthesis and fine resolution fractional frequency synthesis. Unlike the ATX PLL, the fPLL can be used to synthesize frequencies that can drive the core through the FPGA fabric clock networks.

Related Information

- **fPLL** on page 3-11
  For more information on fPLL.
- **fPLL IP** on page 3-13
  For details on implementing the fPLL IP.

Channel PLL (CMU/CDR PLL)

A channel PLL resides locally within each transceiver channel. Its primary function is clock and data recovery in the transceiver channel when the PLL is used in CDR mode. The channel PLLs of channel 1 and 4 can be used as a transmit PLL when reconfigured in CMU mode. The channel PLLs of channel 0, 2, 3, and 5 cannot be reconfigured in CMU mode and therefore cannot be used as a transmit PLL.

Related Information

- **CMU PLL** on page 3-19
  For more information on CMU PLL.
- **CMU PLL IP** on page 3-21
  For information on implementing CMU PLL IP.

Clock Generation Block (CGB)

In Arria 10 devices, there are two types of clock generation blocks (CGBs)

- Master CGB
- Local CGB

Transceiver banks with six transceiver channels have two master CGBs. Master CGB1 is located at the top of the transceiver bank and master CGB0 is located at the bottom of the transceiver bank. Transceiver banks with three channels have only one master CGB. The master CGB divides and distributes bonded clocks to a bonded channel group. It also distributes non-bonded clocks to non-bonded channels across the x6/xN clock network.

Each transceiver channel has a local CGB. The local CGB is used for dividing and distributing non-bonded clocks to its own PCS and PMA blocks.

Related Information

Clock Generation Block on page 3-32
For more information on clock generation block.

Implementing Protocols in Arria 10 Transceivers

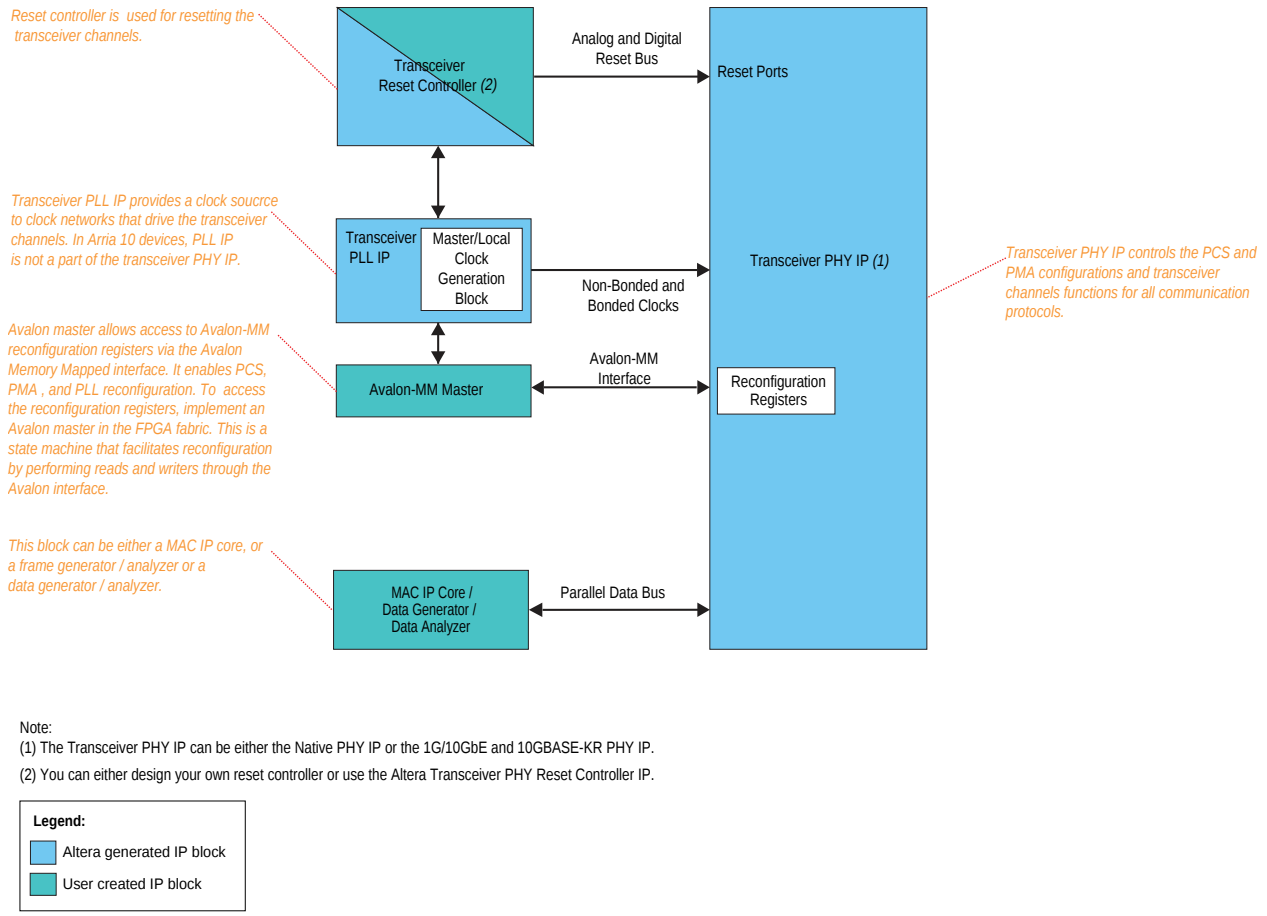
2

2013.12.02

UG-A10XCVR | Subscribe | Send Feedback

Transceiver Design IP Blocks

Figure 2-1: Arria 10 Transceiver Design Fundamental Building Blocks

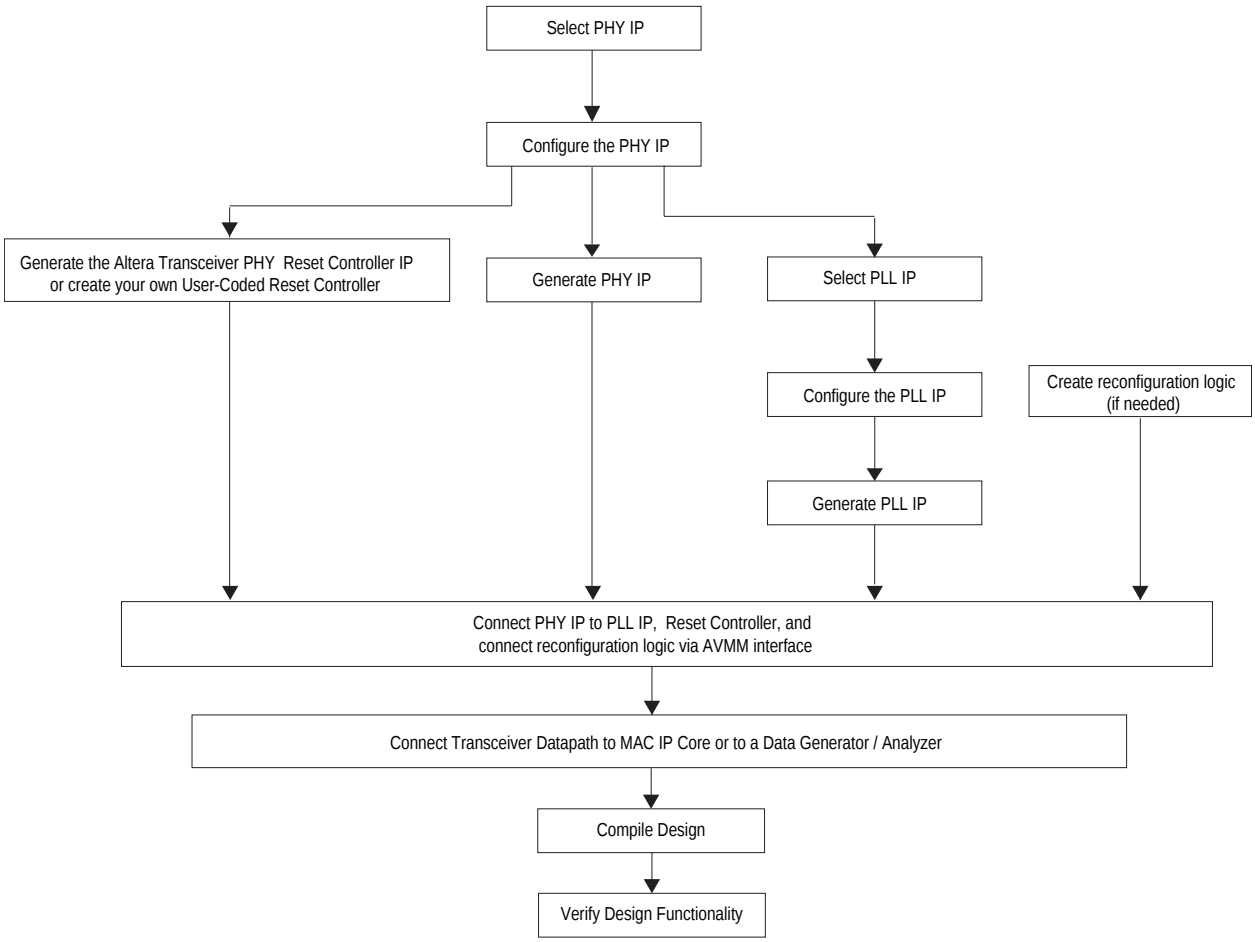

Reset controller is used for resetting the transceiver channels.
Transceiver Reset Controller (2)
Analog and Digital Reset Bus
Reset Ports
Transceiver PLL IP provides a clock soucrce to clock networks that drive the transceiver channels. In Arria 10 devices, PLL IP is not a part of the transceiver PHY IP.
Transceiver PLL IP
Master/Local Clock Generation Block
Non-Bonded and Bonded Clocks
Transceiver PHY IP (1)
Transceiver PHY IP controls the PCS and PMA configurations and transceiver channels functions for all communication protocols.
Avalon master allows access to Avalon-MM reconfiguration registers via the Avalon Memory Mapped interface. It enables PCS, PMA , and PLL reconfiguration. To access the reconfiguration registers, implement an Avalon master in the FPGA fabric. This is a state machine that facilitates reconfiguration by performing reads and writers through the Avalon interface.
Avalon-MM Master
Avalon-MM Interface
Reconfiguration Registers
This block can be either a MAC IP core, or a frame generator / analyzer or a data generator / analyzer.
MAC IP Core / Data Generator / Data Analyzer
Parallel Data Bus
Note:
(1) The Transceiver PHY IP can be either the Native PHY IP or the 1G/10GbE and 10GBASE-KR PHY IP.
(2) You can either design your own reset controller or use the Altera Transceiver PHY Reset Controller IP.
Legend:
Altera generated IP block
User created IP block

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Transceiver Design Flow

Figure 2-2: Transceiver Design Flow


Select PHY IP
Configure the PHY IP
Generate the Altera Transceiver PHY Reset Controller IP or create your own User-Coded Reset Controller
Generate PHY IP
Select PLL IP
Configure the PLL IP
Create reconfiguration logic (if needed)
Generate PLL IP
Connect PHY IP to PLL IP, Reset Controller, and connect reconfiguration logic via AVMM interface
Connect Transceiver Datapath to MAC IP Core or to a Data Generator / Analyzer
Compile Design
Verify Design Functionality

Related Information

Arria 10 Transceiver PHY Design Examples

Note: The design examples on the *alterawiki* page provide useful guidance for developing your own design. However, *alterawiki* is not guaranteed by Altera.

Select and Instantiate PHY IP

Select the appropriate PHY IP to implement your protocol. Refer to the *Arria 10 Transceiver Protocols and PHY IP Support* section to decide which PHY IP to select to implement your protocol.

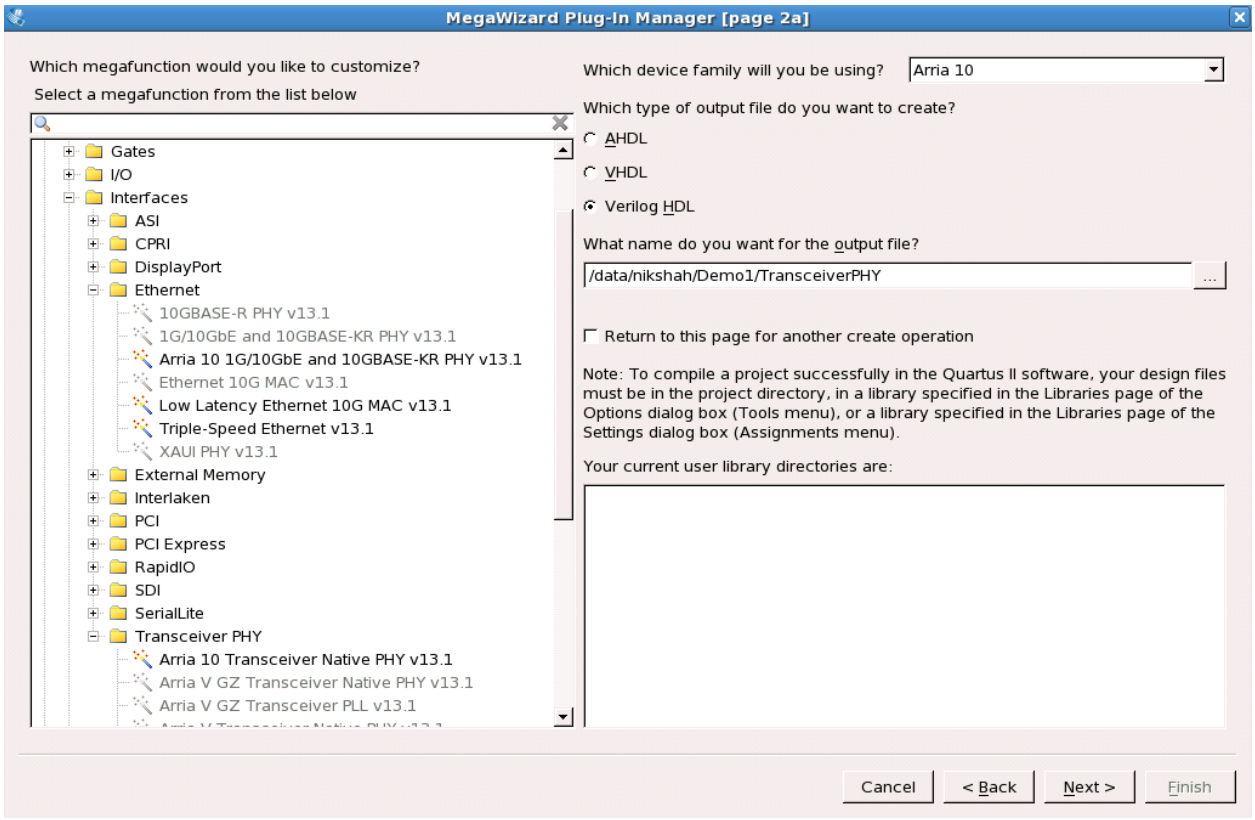
To instantiate a PHY IP:

1. Open the Quartus II software.
2. Click **Tools > MegaWizard Plug-In Manager**.
3. Select **Create a new custom megafunction variation**, then click **Next**.
4. Select **Arria 10** device family.

5. Under **Interfaces**, select the PHY IP you would like to use.
6. Under **Which type of output file do you want to create?**, select **Verilog** or **VHDL** as the hardware description language.
7. In **What name do you want for the output file?**, browse to the location where you want to save your design, and enter a filename.
8. Click **Next**.

The PHY IP GUI window opens.

Figure 2-3: Arria 10 Transceiver PHY Types


MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Gates
I/O
Interfaces
ASI
CPRI
DisplayPort
Ethernet
10GBASE-R PHY v13.1
1G/10GbE and 10GBASE-KR PHY v13.1
Arria 10 1G/10GbE and 10GBASE-KR PHY v13.1
Ethernet 10G MAC v13.1
Low Latency Ethernet 10G MAC v13.1
Triple-Speed Ethernet v13.1
XAUI PHY v13.1
External Memory
Interlaken
PCI
PCI Express
RapidIO
SDI
SerialLite
Transceiver PHY
Arria 10 Transceiver Native PHY v13.1
Arria V GZ Transceiver Native PHY v13.1
Arria V GZ Transceiver PLL v13.1
Which device family will you be using?
Arria 10
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
/data/nikshah/Demo1/TransceiverPHY
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in a library specified in the Libraries page of the Options dialog box (Tools menu), or a library specified in the Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish

Related Information

Arria 10 Transceiver Protocols and PHY IP Support on page 2-7

Configure the PHY IP

Configure the PHY IP by selecting the valid parameters for your design. The valid parameter settings are different for each protocol. Refer to the appropriate protocol's section for selecting valid parameters for each protocol.

Related Information

- Using the Arria 10 Transceiver Native PHY IP on page 2-12
  For information on Native PHY IP.
- Interlaken on page 2-62

- **Gigabit Ethernet (GbE) and GbE with 1588** on page 2-80
- **10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC** on page 2-92
- **10GBASE-KR PHY IP with FEC Option** on page 2-103
- **1G/10 Gbps Ethernet PHY IP Core** on page 2-149
- **PCI Express** on page 2-179
- **CPRI** on page 2-216
- **Using the "Basic" and "Basic with KR FEC" Configurations of Enhanced PCS** on page 2-225
- **Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS** on page 2-232
- **Design Considerations for Data Rates Above 17.4 Gbps Using Arria 10 GT Channels** on page 2-250

Generate PHY IP

After configuring the PHY IP, click the **Finish** button in the MegaWizard ® Plug-In Manager window and generate PHY IP. The Quartus II software generates a <*phy ip instance name*> folder, <*phy ip instance name*>_sim folder, <*phy ip instance name*>.qip file and <*phy ip instance name*>.v file. This <*phy ip instance name*>.v file is the top level design file for the PHY IP and the folders contain lower level design files used for simulation and compilation.

Select PLL IP

Arria 10 devices have three types of PLL IPs:

- Advanced Transmit (ATX) PLL IP
- Fractional PLL (fPLL) IP
- Channel PLL / Clock Multiplier Unit (CMU) PLL IP

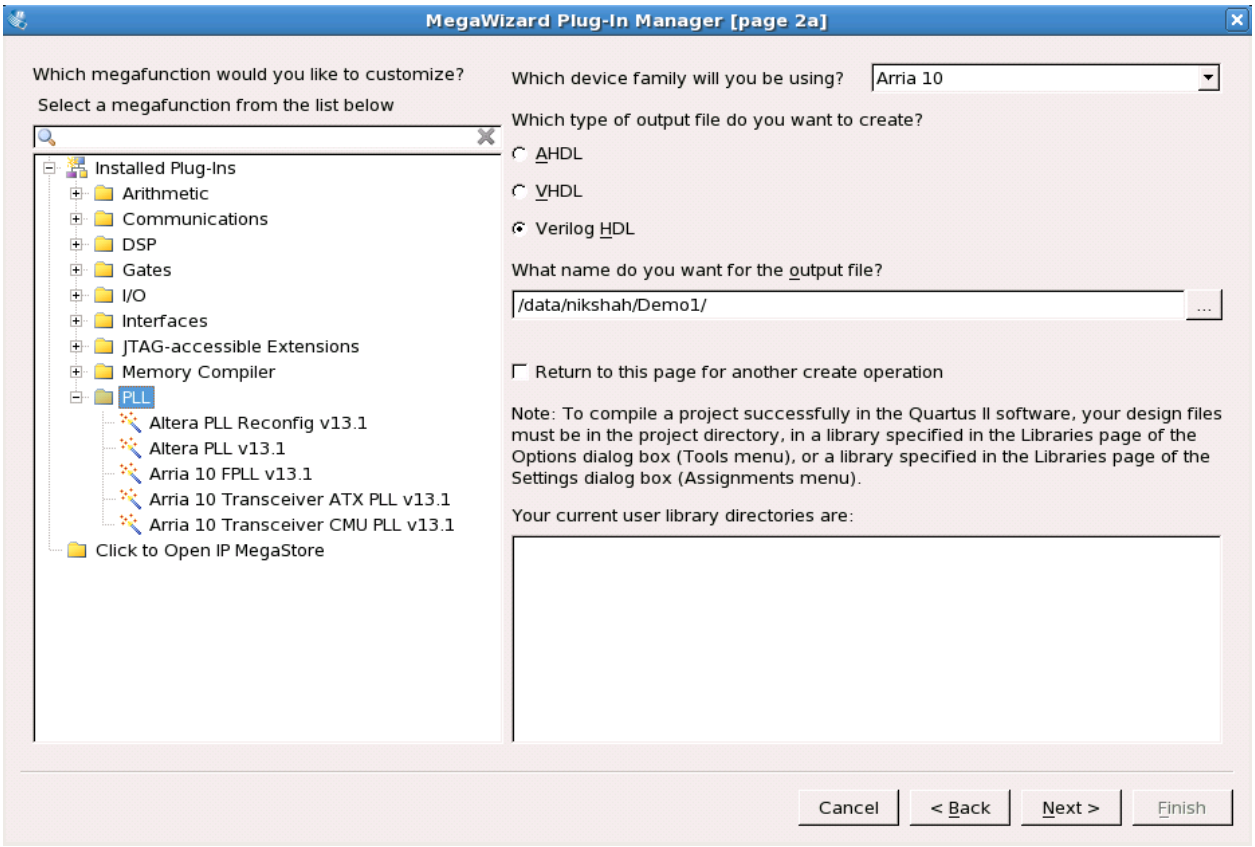
Select the appropriate PLL IP for your design. Refer to the *PLLs and Clock Networks* chapter for detailed information on available PLLs and clock networks.

To instantiate a PLL IP:

1. Open the Quartus II software.
2. Click **Tools > MegaWizard Plug-In Manager**.
3. Select **Create a new custom megafunction variation**, then click **Next**.
4. Select **Arria 10** device family.
5. Select **PLL** in the Installed Plug-Ins tree. Choose the PLL IP (Arria 10 Transceiver ATX PLL, Arria 10 fPLL or Arria 10 Transceiver CMU PLL) you want to instantiate in your design.
6. Under **Which type of output file do you want to create?**, select **Verilog** or **VHDL** as the hardware description language.
7. In **What name do you want for the output file?**, browse to the location where you want to save your design, and enter a filename.
8. Click **Next**.

The PLL IP GUI window opens.

Figure 2-4: Arria 10 Transceiver PLL Types


MegaWizard Plug-In Manager [page 2a]
Which megafunction would you like to customize?
Select a megafunction from the list below
Installed Plug-Ins
Arithmetic
Communications
DSP
Gates
I/O
Interfaces
JTAG-accessible Extensions
Memory Compiler
PLL
Altera PLL Reconfig v13.1
Altera PLL v13.1
Arria 10 FPLL v13.1
Arria 10 Transceiver ATX PLL v13.1
Arria 10 Transceiver CMU PLL v13.1
Click to Open IP MegaStore
Which device family will you be using?
Arria 10
Which type of output file do you want to create?
AHDL
VHDL
Verilog HDL
What name do you want for the output file?
/data/nikshah/Demo1/
Return to this page for another create operation
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in a library specified in the Libraries page of the Options dialog box (Tools menu), or a library specified in the Libraries page of the Settings dialog box (Assignments menu).
Your current user library directories are:
Cancel
< Back
Next >
Finish

Related Information

PLLs on page 3-3

Configure PLL IP

Understand the available PLLs, clock networks and the supported clocking configurations. Configure the PLL IP to achieve adequate data rate for your design.

Related Information

- ATX PLL IP on page 3-6
- fPLL IP on page 3-13
- CMU PLL IP on page 3-21
- Using PLLs and Clock Networks on page 3-40

Generate PLL IP

After configuring the PLL IP, click the **Finish** button in the MegaWizard Plug-In Manager window. This generates the PLL IP. The Quartus ® II software generates a *<pll ip instance name>* folder, *<pll ip instance name>*_sim folder, *<pll ip instance name>*.qip file and *<pll ip instance name>*.v file. The *<pll ip instance*

name>.v file is the top level design file for the PLL IP and the folders contain lower level design files used for simulation and compilation.

Reset Controller

There are two methods to reset the transceivers in Arria 10 devices:

- Using the Altera Transceiver PHY Reset Controller IP Core
- Using your own User-Coded Reset Controller

Related Information

Resetting Transceiver Channels on page 4-1

Create Reconfiguration Logic

Dynamic reconfiguration is the ability to dynamically modify the transceiver channels and PLLs settings during device operation. You need to create an Avalon master in order to access the dynamic reconfiguration registers using the Avalon interface.

The Avalon-MM master enables PCS dynamic switching, PLL and channel reconfiguration. All the PMA parameters such as VOD, differential output voltage swing, and pre-emphasis can be dynamically adjusted through the Avalon-MM reconfiguration registers by the user generated Avalon-MM master.

Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for details on dynamic reconfiguration.

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 6-1

Connect PHY IP to PLL IP and Reset Controller

Connect the PHY IP, PLL IP, and the reset controller. Write the top level module to connect all the IP blocks.

All the I/O ports for each IP can be seen in the <*phy instance name*>.v file.

Refer to the ports tables in the *PLL IP*, *Using the Transceiver Native PHY IP*, and *Resetting Transceiver Channels* chapters for the description of the ports.

Related Information

- **Enhanced PCS and PMA Ports** on page 2-37
- **Standard PCS and PMA Ports** on page 2-52

Connect the Transceiver Datapath to MAC IP Core or to a Data Generator / Analyzer

Connect the transceiver PHY layer design to the Media Access Controller (MAC) IP or to a data generator / analyzer or a frame generator / analyzer. This can be also be a user created IP block.

Compile Design

To compile the transceiver design, ensure all the <*phy_instancename*>.qip files for all the IP blocks generated using the MegaWizard Plug-In Manager are added to the Quartus II project library.

Related Information

Quartus II Incremental Compilation for Hierarchical and Team-Based Design
For compilation details.

Verify Design Functionality

Simulate your design to verify the functionality of your design. Refer to *Simulating the Native Transceiver PHY IP Core* section for more details.

Related Information

- **Quartus II Handbook - Volume 3: Verification**
  For information on design simulation and verification.
- **Simulating the Transceiver Native PHY IP Core** on page 2-256

Arria 10 Transceiver Protocols and PHY IP Support

Table 2-1: Arria 10 Transceiver Protocols and PHY IP Support

Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
PCIe Gen3 x1, x2, x4, x8	Native PHY IP (PIPE)[19]	Standard and Gen3	PIPE Gen3	PCIe PIPE Gen3 x1 PCIe PIPE Gen3 x8
PCIe Gen2 x1, x2, x4, x8	Native PHY IP (PIPE) [19]	Standard	PIPE Gen2	PCIe PIPE Gen2 x1 PCIe PIPE Gen2 x8
PCIe Gen1 x1, x2, x4, x8	Native PHY IP (PIPE) [19]	Standard	PIPE Gen1	User created
1000BASE-X Gigabit Ethernet	Native PHY IP	Standard	GbE	GIGE - 1.25 Gbps
1000BASE-X Gigabit Ethernet with 1588	Native PHY IP	Standard	GbE 1588	GIGE - 1.25 Gbps 1588

(17) For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(18) For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(19) PCI Express Hard IP is also available as a MegaCore® function.
(20) The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
(21) Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
(22) PCS-Direct support will be available in a future release of the Quartus II software.
(23) A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
(24) XAUI PHY IP will be available in a future release of the Quartus II software.

Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
10GBASE-R	Native PHY IP	Enhanced	10GBASE-R	10GBASE-R
10GBASE-R 1588	Native PHY IP	Enhanced	10GBASE-R 1588	10GBASE-R 1588
10GBASE-R with KR FEC	Native PHY IP	Enhanced	10GBASE-R w/ KR FEC	10GBASE-R w/KR FEC
10GBASE-KR and 1000BASE-X	1G/10GbE and 10GBASE-KR PHY IP[20]	Standard and Enhanced	Not applicable	BackPlane_wo_1588 LineSide (optical) LineSide(optical)_1588
40GBASE-R/100GBASE-R	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
40GBASE-R with FEC/ 40GBASE-KR4 [21]	Native PHY IP	Enhanced	Basic w/KR FEC	User created
100GBASE-R via CAUI-4/ CPPI-4/BP-4	Native PHY IP	PCS-Direct[22] / Enhanced PCS (low latency mode)	Basic (Enhanced PCS)	User created
100GBASE-R via CAUI	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
100GBASE-R via CAUI with FEC	Native PHY IP	Enhanced	Basic w/KR FEC	User created
XAUI	XAUI PHY IP [24]	Standard Soft PCS	Not applicable	Not applicable
SPAUI	Native PHY IP	Standard and Enhanced	Basic/Custom (Standard PCS) Basic (Enhanced PCS)	User created

[17] For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
[18] For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
[19] PCI Express Hard IP is also available as a MegaCore® function.
[20] The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
[21] Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
[22] PCS-Direct support will be available in a future release of the Quartus II software.
[23] A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
[24] XAUI PHY IP will be available in a future release of the Quartus II software.

Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
DDR XAUI	Native PHY IP	Standard and Enhanced	Basic/Custom (Standard PCS) Basic (Enhanced PCS)	User created
Interlaken (CEI-6G/11G) [23]	Native PHY IP	Enhanced	Interlaken	Interlaken 10x12.5Gbps Interlaken 6x10.3Gbps Interlaken 1x6.25Gbps
OTU-4 (100G) via OTL4.4/ CEI-25G/28G VSR/SR	Native PHY IP	PCS-Direct[22] Enhanced PCS (low latency mode)	Basic (Enhanced PCS)	User created
OTU-4 (100G) via OTL4.10/ OIF SFI-S	Native PHY IP	Enhanced	SFI-S	User created
OTU-3 (40G) via OTL3.4/ OIF SFI-5.2/SFI-5.1	Native PHY IP	Enhanced	SFI-S	User created
OTU-2 (10G) via SFP+/SFF-8431/CEI-11G	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
OTU-2 (10G) via OIF SFI-5.1s	Native PHY IP	Enhanced	SFI-S	User created
OTU-1 (2.7G)	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
SONET/SDH STS-768/STM-256 (40G) via OIF SFI-5.2/ STL256.4	Native PHY IP	Enhanced	SFI-S	User created
SONET/SDH STS-768/STM-256 (40G) via OIF SFI-5.1	Native PHY IP	Enhanced	SFI-S	User created

(17) For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(18) For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(19) PCI Express Hard IP is also available as a MegaCore® function.
(20) The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
(21) Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
(22) PCS-Direct support will be available in a future release of the Quartus II software.
(23) A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
(24) XAUI PHY IP will be available in a future release of the Quartus II software.

Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
SONET/SDH STS-192/STM-64 (10G) via SFP+/SFF-8431/CEI-11G	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
SONET/SDH STS-192/STM-64 (10G) via OIF SFI-5.1s/ SxI-5/SFI-4.2	Native PHY IP	Enhanced	SFI-S	User created
SONET STS-96 (5G) via OIF SFI-5.1s	Native PHY IP	Enhanced	SFI-S	User created
SONET/SDH STS-48/STM-16 (2.5G) via SFP/TFI-5.1	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
SONET/SDH STS-12/STM-4 (0.622G) via SFP/TFI-5.1	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
Intel QPI 1.1/2.0	Native PHY IP	PCS-Direct [22]	Not Available	User created
10G SDI	Native PHY IP	Enhanced	10G SDI	User created
SD-SDI/HD-SDI/3G-SDI	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
Vx1	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
DisplayPort	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
1.25G/ 2.5G/ 10G GPON/ EPON	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
2.5G/1.25G GPON/EPON	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
16G/10G Fibre Channel	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created

(17) For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(18) For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(19) PCI Express Hard IP is also available as a MegaCore® function.
(20) The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
(21) Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
(22) PCS-Direct support will be available in a future release of the Quartus II software.
(23) A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
(24) XAUI PHY IP will be available in a future release of the Quartus II software.

Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
8G/4G/2G/1G Fibre Channel	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
EDR Infiniband x1, x4, x12	Native PHY IP	PCS-Direct [22] Enhanced (low latency mode)	Basic (Enhanced PCS)	User created
FDR/FDR-10 Infiniband x1, x4, x12	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
SDR/DDR/QDR Infiniband x1, x4, x12	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
CPRI 6.0 10.1376 Gbps	Native PHY IP	Enhanced	10GBASE-R 1588	10GBASE-R 1588
CPRI 4.2/OBSAI RP3 v4.2	Native PHY IP	Standard	CPRI (Auto) / CPRI (Manual)	CPRI 9.8Gbps Auto Mode CPRI 9.8 Gbps Manual Mode
SRIO 2.2/1.3	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
SAS 3.0	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
SATA 3.0/2.0/1.0 and SAS 2.0/1.0	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
HiGig/HiGig+/HiGig2/ HiGig2+	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
JESD204B	Native PHY IP	Enhanced	Basic (Enhanced PCS)	User created
JESD204A	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created

(17) For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(18) For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(19) PCI Express Hard IP is also available as a MegaCore® function.
(20) The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
(21) Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
(22) PCS-Direct support will be available in a future release of the Quartus II software.
(23) A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
(24) XAUI PHY IP will be available in a future release of the Quartus II software.

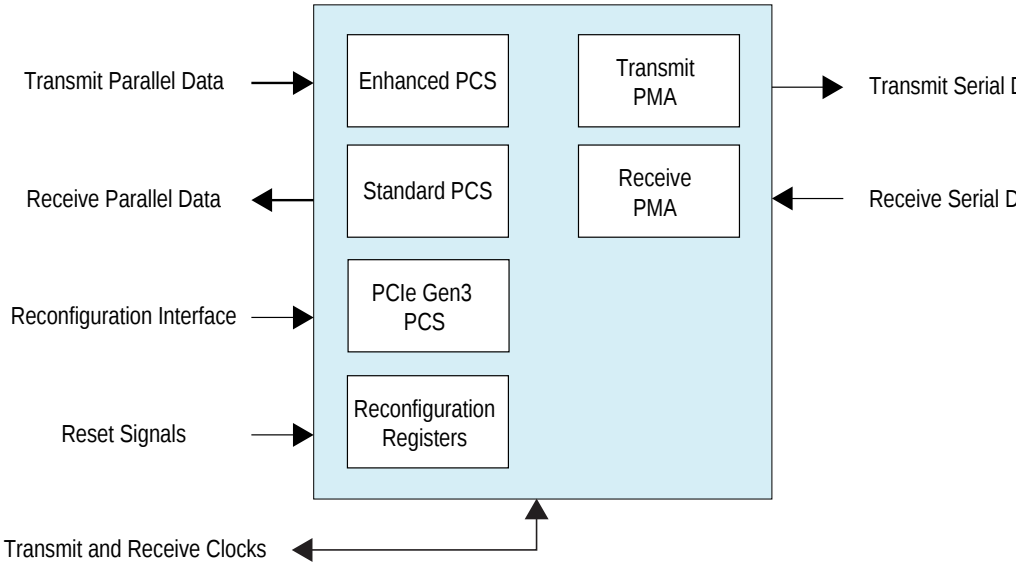
Protocol	Transceiver IP	PCS Support	Transceiver Configuration Rule[17]	Protocol Preset [18]
ASI	Native PHY IP	Standard	Basic/Custom (Standard PCS)	User created
SPI 5 (100G) SPI 5 (50G)	Native PHY IP	Enhanced	SFI-S	User created
Custom and other procols	Native PHY IP	Standard and Enhanced	Basis/Custom (Standard PCS) Basic (Enhanced PCS) Basic/Custom with Rate Match (Standard PCS)	User created

Using the Arria 10 Transceiver Native PHY IP

This section describes use of the Altera-provided Arria 10 Transceiver Native PHY IP core. This IP core provides direct access to Arria 10 transceiver features. You can enable the Standard and/or Enhanced PCS datapaths. If you enable both the Standard and Enhanced PCS datapaths, you can use the Reconfiguration Interface to switch between them without device power down. You can enable Gen3 PCS by selecting the Gen3 PIPE transceiver configuration rule in standard PCS. Similarly, you can customize the Transceiver Native PHY IP by specifying various IP parameters.

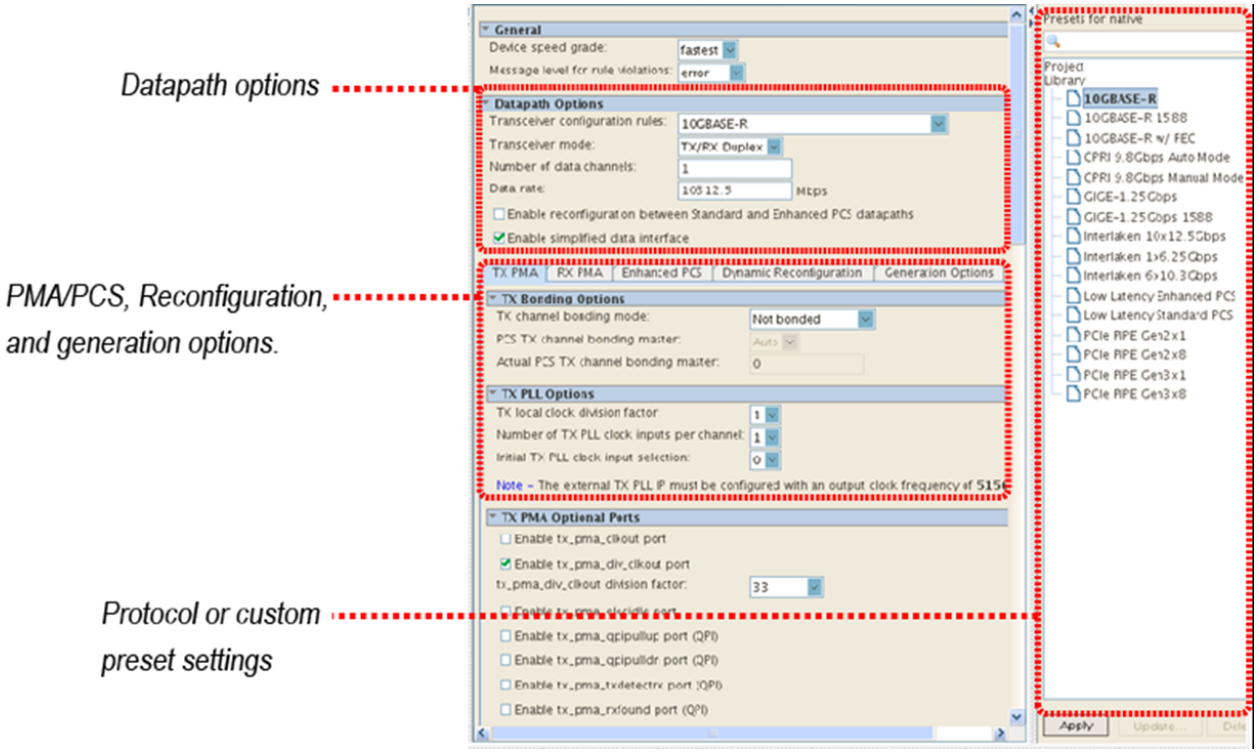
(17) For more information on Transceiver Configuration Rules, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(18) For more information on Protocol Presets, refer to **Using the Arria 10 Transceiver Native PHY IP** on page 2-12.
(19) PCI Express Hard IP is also available as a MegaCore® function.
(20) The 1G/10GbE and 10GBASE-KR PHY IP includes the necessary Soft IP for link training, auto speed negotiation, and sequencer functions.
(21) Needs a user created Soft-IP for link training, auto speed negotiation, and sequencer functions.
(22) PCS-Direct support will be available in a future release of the Quartus II software.
(23) A Soft PCS bonding IP design example will be available in a future release of the Quartus II software.
(24) XAUI PHY IP will be available in a future release of the Quartus II software.

Figure 2-5: Transceiver Native PHY IP Top-Level Interfaces and Functional Blocks



Specify IP parameters by clicking **Tools** > **MegaWizard Plug-In Manager** and selecting your IP core variation. To quickly specify appropriate initial settings for your configuration, select a **Preset** matching your configuration. Select **Transceiver configuration rules** to report an error for any parameters incompatible with the specified PCS and PMA.

Figure 2-6: Transceiver Native PHY IP GUI



Note: The Quartus II software version 13.1 does not perform legality checks for PCS to FPGA fabric speeds. Although the Quartus II software version 13.1 checks the Transceiver data rate, the PCS to FPGA fabric may be set too high if the PCS/PMA interface width is too small for high data rates. Later versions of the Quartus II software perform PCS interface legality checking.

Related Information

- **Configure the PHY IP** on page 2-3
- **Interlaken** on page 2-62
- **Gigabit Ethernet (GbE) and GbE with 1588** on page 2-80
- **10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC** on page 2-92
- **10GBASE-KR PHY IP with FEC Option** on page 2-103
- **1G/10 Gbps Ethernet PHY IP Core** on page 2-149
- **PCI Express** on page 2-179
- **CPRI** on page 2-216
- **Using the "Basic" and "Basic with KR FEC" Configurations of Enhanced PCS** on page 2-225
- **Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS** on page 2-232

- **Design Considerations for Data Rates Above 17.4 Gbps Using Arria 10 GT Channels** on page 2-250
- **PMA Parameters** on page 2-17

General and Datapath Parameters

You can customize your instance of the Transceiver Native PHY IP by specifying parameter values. The GUI organizes the parameters into the following sections for each functional block and feature:

- General and Datapath Options
- TX PMA
- RX PMA
- Enhanced PCS
- Standard PCS
- Dynamic Reconfiguration

Table 2-2: General and Datapath Options

Parameter	Value	Description
Device speed grade	**fastest**	Specifies the required speed grade. This information is used for data rate validation.
Message level for rule violations	**error** **message**	Specifies the messaging level for parameter rule violations. Selecting **error** causes all rule violations to prevent IP generation. Selecting **warning** displays all rule violations as warnings in the message window and allows IP generation despite the violations.
Transceiver mode	**TX/RX Duplex** **TX Simplex** **RX Simplex**	Specifies the operational mode of the transceiver. • **TX/RX Duplex**: Specifies a single channel that supports both transmit and receive capabilities. • **TX Simplex**: Specifies a single channel that supports only transmission. • **RX Simplex**: Specifies a single channel that supports only reception. The default is **TX/RX Duplex**.
Number of data channels	**1**–<*n*>	Specifies the number of transceiver channels to be implemented. The maximum number of channels available, (<*n*>), depends on the package you select. The default value is **1**.
Data rate	<*valid Arria 10 Transceiver data rate*>	Specifies the data rate in megabits per second (Mbps). The default value is 1250 Mbps.

Parameter	Value	Description
Enable reconfiguration between Standard and Enhanced PCSs	**On/Off**	When you turn this option on, you can preconfigure both the Standard and Enhanced PCS datapaths and dynamically reconfigure between them The default value is **Off**.
Enable simplified data interface	**On/Off**	By default, all 128-bits are ports for the `tx_parallel_data` and `rx_parallel_data` buses, regardless of the FPGA fabric width specified. You must understand the mapping of data and control signals within the interface. When you turn on this option, the Transceiver Native PHY IP presents a simplified data and control interface between the FPGA fabric and transceiver. Only the sub-set of the 128-bits that are active for a particular FPGA fabric width are ports. The default value is **Off**.

Table 2-3: Transceiver Configuration Rule Parameters

Transceiver Configuration Setting	Description
Basic/Custom (Standard PCS)	Enforces a standard set of rules within the Standard PCS. Select these rules to implement custom protocols requiring blocks within the Standard PCS or protocols not covered by the other configuration rules.
Basic/Custom w/Rate Match (Standard PCS)	Enforces a standard set of rules including rules for the Rate Match FIFO within the Standard PCS. Select these rules to implement custom protocols requiring blocks within the Standard PCS or protocols not covered by the other configuration rules.
CPRI (Auto)	Enforces rules required by the CPRI protocol. The receiver word aligner mode is set to **Auto**. In **Auto** mode, the word aligner is set to deterministic latency.
CPRI (Manual)	Enforces rules required by CPRI protocol. The receiver word aligner mode is set to **Manual**. In **Manual** mode, logic in the FPGA fabric controls the word aligner.
GbE	Enforces rules that the 1 Gbps Ethernet (1 GbE) protocol requires.
GbE 1588	Enforces rules for the 1 GbE protocol with support for Precision time protocol (PTP) as defined in the *IEEE 1588 Standard* .
Gen1 PIPE	Enforces rules for a Gen1 PCIe® PIPE interface that you can connect to soft MAC and Data Link Layer.
Gen2 PIPE	Enforces rules for a Gen2 PCIe PIPE interface that you can connect to soft MAC and Data Link Layer.
Gen3 PIPE	Enforces rules for a Gen3 PCIe PIPE interface that you can connect to soft MAC and Data Link Layer.

Transceiver Configuration Setting	Description
Basic (Enhanced PCS)	Enforces a standard set of rules within the Enhanced PCS. Select these rules to implement protocols requiring blocks within the Enhanced PCS or protocols not covered by the other configuration rules.
Interlaken	Enforces rules required by the Interlaken protocol.
SFIS	Enforces rules required by the SFIS protocol.
10G SDI	Enforces rules required by the SDI 10G protocol.
10GBASE-R	Enforces rules required by the 10GBASE-R protocol.
10GBASE-R 1588	Enforces rules required by the 10GBASE-R protocol when you enable 1588.
10GBASE-R w/KR FEC	Enforces rules required by the 10GBASE-R protocol when you enable the KR FEC block enabled.
Basic w/KR FEC	Enforces a standard set of rules required by the Enhanced PCS when you enable the KR FEC block. Select this rule to implement custom protocols requiring blocks within the Enhanced PCS or protocols not covered by the other configuration rules.

PMA Parameters

You can specify values for the following types of PMA parameters:

- TX Bonding Options
- TX PLL Options
- TX PMA Optional Ports
- RX CDR Options
- RX PMA Optional Ports

Table 2-4: TX PMA Bonding Options

Parameter	Value	Description
TX channel bonding mode	**Not bonded** **PMA bonding** **PMA /PCS bonding**	Selects the bonding mode to be used for the channels specified. Bonded clocks use a single TX PLL to generate a clock that drives multiple channels, reducing channel-to-channel skew. The following options are available: **Non bonded**: In a non-bonded configuration, only the high speed serial clock is routed from the transmitter PLL to the transmitter channel. The low speed parallel clock is generated by the local clock generation block (CGB) present in the transceiver channel. For non-bonded configurations, because the channels are not related to each other and the feedback path is local to the PLL, the skew between channels cannot be calculated. **PMA bonding**: In a PMA bonded configuration, the high speed serial clock is routed from the same transmitter PLL to all the transmitter channels that require bonding. The master CGB generates both the high and slow speed clocks. **PMA/PCS bonding**: In a PMA/PCS bonded configuration, both the high speed serial and low speed parallel clocks are routed from the transmitter PLL to the transmitter channel. In this case, the local CGB in each channel is bypassed and the parallel clocks generated by the Master CGB are used to clock the network. The master CGB generates both the high and slow speed clocks. a master channel generates the PCS control signals and distributes to other channels through a control plane block. The default value is **Not bonded**.
PCS TX channel bonding master	**Auto, 0, 1, 2, 3**	Specifies the master PCS channel for PCS bonded configurations. Each Transceiver Native PHY IP instance configured with bonding must specify a bonding master. If you select **Auto**, the Transceiver Native PHY IP automatically selects a recommended channel. The default value is **Auto**. Refer to the *PLLs and Clock Networks* chapter for more information about the TX channel bonding master.
Actual PCS TX channel bonding master	**0, <*no of channels*> - 1**	This parameter is automatically populated based on your selection for the**PCS TX channel bonding master** parameter. Indicates the selected master PCS channel for PCS bonded configurations.

Table 2-5: TX PLL Options

Parameter	Value	Description
TX local clock division factor	**1, 2, 4, 8**	Specifies the value of the divider available in the transceiver channels to divide the TX PLL output clock to generate the correct frequencies for the parallel and serial clocks.
Number of TX PLL clock inputs per channel	**1, 2, 3**	Specifies the number of clock inputs per PLL channel. Use this parameter when you plan to dynamically switch between TX PLL clock sources. Up to 4 input sources are possible.
Initial TX PLL clock input selection	**0-<*no of TX PLL clock inputs*>-1**	Specifies the initially selected TX PLL clock input. This parameter is necessary when you plan to switch between multiple TX PLL clock inputs.

Table 2-6: TX PMA Optional Ports

Parameter	Value	Description
Enable tx_pma_clkout port	**On/Off**	Enables the optional `tx_pma_clkout` output clock. This is the low speed parallel clock from the TX PMA. The source of this clock is the deserializer. It is driven by the PCS/PMA interface block.
Enable tx_pma_div_clkout port	**On/Off**	Enables the optional `tx_pma_div_clkout` output clock. This clock is generated by the serializer. You can use this to drive core logic, to drive the PCS-to-fabric, or both. If you specify a **tx_pma_div_clkout division factor** of 1 or 2, this clock output is derived from the PMA parallel clock. If you specify a **tx_pma_div_clkout division factor** of 33, 40, or 66, this clock is derived from the PMA serial clock. This clock is commonly used when the interface to the TX FIFO runs at a different rate than the PMA parallel clock frequency, such as 66:40 applications.
tx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**	Specifies the division factor for the `tx_pma_div_clkout` output clock when this port is enabled.
tx_pma_elecidle port	**On/Off**	Enables the `tx_pma_elecidle` port. When you assert this port, the transmitter is forced into an electrical idle condition. This port has no effect when the transceiver is configured for PCI Express.
Enable tx_pma_qpipullup port (QPI)	**On/Off**	Enables the `tx_pma_qpipullup` control input port. Use this port only for Quick Path Interconnect (QPI) applications.
Enable tx_pma_qpipulldn port (QPI)	**On/Off**	Enables the `tx_pma_qpipulldn` control input port. Use this port only for QPI applications.

Parameter	Value	Description
Enable tx_pma_txdetectrx port (QPI)	**On/Off**	Enables the `tx_pma_txdetectrx` control input port. The receiver detect block in TX PMA detects the presence of a receiver at the other end of the channel. After receiving `tx_pma_txdetectrx` request the receiver detect block initiates the detection process. Use this port only in QPI applications.
Enable tx_pma_rxfound port (QPI)	**On/Off**	Enables the `tx_rxfound` status output port. The receiver detect block in TX PMA detects the presence of a receiver at the other end by using `tx_pma_txdetectrx` input. The `tx_pma_rxfound` reports the status of the detection operation. Use this port only in QPI applications.

Table 2-7: RX PMA Parameters

Parameter	Value	Description
Number of CDR reference clocks	**1-5**	Specifies the number of CDR reference clocks. Up to 5 sources are possible. The default value is **1**.
Selected CDR reference clock	**0 - <*number of CDR reference clocks*>**	Specifies the initial CDR reference clock. The parameter **Number of CDR reference clocks** determines the available CDR references available. used. The default value is **0**.
Selected CDR reference clock frequency	*<data rate dependent>*	Specifies the CDR reference clock frequency. This value depends on the data rate specified.
PPM detector threshold	**62.5** , **100**, **125**, **200**, **250**, **300**, **500**, **1000**	Specifies the PPM threshold for the CDR. If the PPM threshold value you select here is exceeded, the CDR loses lock. The default value is **1000**.
Decision feedback equalization mode	Disabled Fixed tap Floating tap	Specifies the operating mode for the decision feedback equalization (DFE) block in the RX PMA. The default value is **Disabled**.

Table 2-8: RX PMA Optional Ports

Parameters	Value	Description
Enable rx_pma_clkout port	**On/Off**	Enables the optional `rx_pma_clkout` output clock. This port is the recovered parallel clock from the RX clock data recover (CDR).

Parameters	Value	Description
Enable rx_pma_div_clkout port	**On/Off**	Enables the optional `rx_pma_div_clkout` output clock. The deserializer generates this clock. Use this to drive core logic, to drive the RX PCS-to-fabric interface, or both. If you specify a **rx_pma_div_clkout division factor** of 1 or 2, this clock output is derived from the PMA parallel clock. If you specify a **rx_pma_div_clkout division factor** of 33, 40, or 66, this clock is derived from the PMA serial clock. This clock is commonly used when the interface to the RX FIFO runs at a different rate than the PMA parallel clock frequency, such as 66:40 applications.
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**	Specifies the division factor for the `rx_pma_div_clkout` output clock when this port is enabled.
Enable rx_pma_div_clkout division factor	**On/Off**	Specifies the division factor for the `rx_pma_div_clkout` clock signal.This parameter is disabled when you turn Off the **Enable rx_pma_div_clkout port**.
Enable rx_pma_clkslip port	**On/Off**	Enables the optional `rx_pma_clkslip` control input port. A rising edge on this signal causes the RX serializer to slip the serial data by one clock cycle, or 2 unit intervals (UI).
Enable rx_pma_qpipulldn port (QPI)	**On/Off**	Enables the `rx_pma_qpipulldn` control input port. Use this port only for QPI applications.
Enable rx_is_lockedtodata port	**On/Off**	Enables the optional `rx_is_lockedtodata` status output port. This signal indicates that the RX CDR is currently in lock to data mode or is attempting to lock to the incoming data stream. This is an asynchronous output signal.
Enable rx_is_lockedtoref port	**On/Off**	Enables the optional `rx_is_lockedtoref` status output port. This signal indicates that the RX CDR is currently locked to the CDR reference clock. This is an asynchronous output signal.
Enable rx_set_lockedtodata port and rx_set_lockedtoref ports	**On/Off**	Enables the optional `rx_is_lockedtodata` and `rx_set_lockedtoref` control input ports. You can use these control ports to manually control the lock mode of the RX CDR. These are asynchronous input signals.
Enable rx_seriallpbken port	**On/Off**	Enables the optional `rx_seriallpbken` control input port. When you assert this port `rx_seriallpbken`, the TX to RX serial loopback path is enabled. This is an asynchronous input signal.
Enable PRBS verifier control and status port	**On/Off**	Enables the optional `rx_prbs_err`, `rx_prbs_clr`, and `rx_prbs_done` control ports. These ports control and collect status from the internal PRBS verifier.

Related Information

PLLs and Clock Networks on page 3-1

Enhanced PCS Parameters

This section defines parameters available in the Transceiver Native PHY GUI to customize the individual blocks in the Enhanced PCS.

You can implement the following protocols using the Enhanced PCS:

- Ethernet 10GBASE-R and 10GBASE-KR
- Interlaken
- SFI-S and SFI-5.2
- 10G-SDI
- IEEE 1588

The following tables describe the parameters available. If you specify an industry-standard protocol in the GUI, the Transceiver Native PHY IP prints error messages if the settings specified violate the protocol standard.

Table 2-9: Enhanced TX FIFO Parameters

Note: For detailed descriptions of the optional ports that you can enable or disable, refer to the **Enhanced PCS and PMA Ports** on page 2-37.

Parameter	Range	Description
Enable RX/TX FIFO double width mode	**On/Off**	Enables the double width mode for the RX and TX FIFOs. You can use double width mode to run FPGA fabric at half the frequency of the PCS.
TX FIFO Mode	**Phase Compensation** **Register** **Interlaken** **Basic**	Specifies one of the following modes: • **Phase Compensation**: The TX FIFO compensates for the clock phase difference between the read `tx_clkout` and write `tx_coreclkin` or `tx_clkout` clocks. • **Register** : The TX FIFO is bypassed. `tx_parallel_data`, `tx_control` and `tx_enh_data_valid` are registered at the FIFO output. You must control `tx_enh_data_valid` based on gearbox ratio to avoid gearbox underflow or overflow conditions. • **Interlaken** : The TX FIFO acts as an elastic buffer. In this mode, there are additional signals to control the data flow into the FIFO. Therefore, the FIFO write clock frequency does not have to be the same as the read clock frequency. You can control writes to the FIFO with `tx_enh_data_valid`. By monitoring the FIFO flags, you can avoid the FIFO full and empty conditions. The Interlaken frame generator controls reads. • **Basic**: The TX FIFO acts as an elastic buffer to control the input data flow , using `tx_enh_data_valid`. The gearbox data valid flag controls the FIFO read enable.

Parameter	Range	Description
TX FIFO partially full threshold	**10, 11, 12, 13, 14, 15**	Specifies the partially full threshold for the Enhanced PCS TX FIFO. Enter the value at which you want to the TX FIFO to flag a partially full status. `tx_enh_fifo_pfull` is synchronous to `tx_coreclkin`.
Enhanced PCS/ PMA interface width	**32, 40, 64**	Specifies the data interface width between the Enhanced PCS and the transceiver PMA.
FPGA fabric/ Enhanced PCS interface width	**32, 40, 50, 64, 66, 67**	Specifies the FPGA fabric to transceiver PCS interface width between the Enhanced PCS and the FPGA Fabric. The 66-bit FPGA fabric/PCS interface width uses 64-bits from the TX and RX parallel data and the lower 2-bits from the control bus. The 67-bit FPGA fabric/PCS interface width uses the 64-bits from the TX and RX parallel data and the lower 3-bits from the control bus.
TX FIFO partially empty threshold	**1, 2, 3, 4, 5**	Specifies the partially empty threshold for the Enhanced PCS TX FIFO. Enter the value at which you want TX FIFO to flag a partially empty status.
Enable tx_enh_ fifo_full port	**On/Off**	Enables the `tx_enh_fifo_full` port.
Enable tx_enh_ fifo_pfull port	**On/Off**	Enables the `tx_enh_fifo_pfull` port.
Enable tx_enh_ fifo_empty port	**On/Off**	Enables the `tx_enh_fifo_empty` port.
Enable tx_enh_ fifo_pempty port	**On/Off**	Enables the `tx_enh_fifo_pempty` port.
Enable tx_enh_ fifo_cnt port	**On/Off**	Enables the `tx_enh_fifo_cnt` port.

Table 2-10: Enhanced RX FIFO Parameters

Parameter	Range	Description
RX FIFO Mode	**Phase Compensation** **Register** **Interlaken** **10GBASE-R** **Basic**	Specifies one of the following modes for Enhanced PCS RX FIFO: • **Phase Compensation**: This mode compensates for the clock phase difference between the read `rx_coreclkin` or `rx_clkout` and write clock `rx_clkout`. • **Register** : The TX FIFO is bypassed. `rx_parallel_data`, `rx_control`, and `rx_enh_data_valid` are registered at the FIFO output. • **Interlaken**: Select this mode for the Interlaken protocol. To implement the deskew process, you must implement a FSM that controls the FIFO operation based on FIFO flags. In this mode the FIFO acts as an elastic buffer. • **10GBASE-R**: In this mode, data passes through the FIFO after block lock is achieved. Idles/OS (Ordered Sets) are deleted and Idles are inserted to compensate for the clock difference between the RX PMA clock and the fabric clock of +/- 100 ppm for a maximum packet length of 64000 bytes. • **Basic:** In this mode, the RX FIFO acts as an elastic buffer. The gearbox data valid flag controls the FIFO read enable. You can monitor the `rx_enh_fifo_pfull` and `rx_enh_fifo_empty` flags to determine whether or not to read from the FIFO.
RX FIFO partially full threshold	0-31	Specifies the partially full threshold for the Enhanced PCS RX FIFO. The default value is 23.
RX FIFO partially empty threshold	0-31	Specifies the partially empty threshold for the Enhanced PCS RX FIFO. The default value is 2.
Enable RX FIFO alignment word deletion (Interlaken)	**On/Off**	When you turn this option on, all alignment words (sync words) , including the first sync word, are removed after frame synchronization is achieved. If you enable this option, you must also enable control word deletion.
Enable RX FIFO control word deletion (Interlaken)	**On/Off**	When you turn this option on, Interlaken control word removal is enabled. When the Enhanced PCS RX FIFO is configured in **Interlaken** mode, enabling this option removes all control words after frame synchronization is achieved. Enabling this option requires that you also enable alignment word deletion.
Enable rx_enh_data_valid port	**On/Off**	Enables the `rx_enh_data_valid` port.
Enable rx_enh_fifo_full port	**On/Off**	Enables the `rx_enh_fifo_full` port.

Parameter	Range	Description
Enable rx_enh_fifo_pfull port	**On/Off**	Enables the active high `rx_enh_fifo_pfull` port.
Enable rx_enh_fifo_empty port	**On/Off**	Enables the active high `rx_enh_fifo_empty` port.
Enable rx_enh_fifo_pempty port	**On/Off**	Enables the `rx_enh_fifo_pempty` port.
Enable rx_enh_fifo_cnt port	**On/Off**	Enables the optional `rx_enh_fifo_cnt` status output port.
Enable rx_enh_fifo_del port (10GBASE-R)	**On/Off**	Enables the optional `rx_enh_del_cnt` status output port.
Enable rx_enh_fifo_insert port (10GBASE-R)	**On/Off**	Enables the `rx_enh_fifo_insert` port.
Enable rx_enh_fifo_rd_en port (Interlaken)	**On/Off**	Enables the `rx_enh_fifo_rd_en` input port.
Enable rx_enh_fifo_align_val port (Interlaken)	**On/Off**	Enables the `rx_enh_fifo_align_val` status output port.
Enable rx_enh_fifo_align_clr port (Interlaken)	**On/Off**	Enables the `rx_enh_fifo_align_clr` input port.

Table 2-11: Interlaken Frame Generator Parameters

Parameter	Range	Description
Enable Interlaken frame generator	**On/Off**	Enables the frame generator block of the Enhanced PCS.
Frame generator metaframe length	5-8192	Specifies the metaframe length of the frame generator. This metaframe length includes 4 framing control words created by frame generator.
Enable frame burst	**On/Off**	Enables frame generator burst. This determines whether the frame generator reads data from the TX FIFO based on the input of port `tx_enh_frame_burst_en`.
Enable tx_enh_frame port	**On/Off**	Enables the `tx_enh_frame` status output port.
Enable tx_enh_frame_diag_status port	**On/Off**	Enables the `tx_enh_frame_diag_status` 2-bit input port.

Parameter	Range	Description
Enable tx_enh_frame_burst_en port	**On/Off**	Enables the `tx_enh_frame_burst_en` input port.

Table 2-12: Interlaken Frame Synchronizer Parameters

Parameter	Range	Description
Enable Interlaken frame synchronizer	**On/Off**	When you turn this option on, the Enhanced PCS frame synchronizer is enabled.
Frame synchronizer metaframe length	5-8192	Specifies the metaframe length of the frame synchronizer
Enable rx_enh_frame port	**On/Off**	Enables the `rx_enh_frame` status output port.
Enable rx_enh_frame_lock port	**On/Off**	Enables the `rx_enh_frame_lock` output port.
Enable rx_enh_frame_diag_status port	**On/Off**	Enables the `rx_enh_frame_diag_status` port.

Table 2-13: Interlaken CRC32 Generator and Checker Parameters

Parameter	Range	Description
Enable Interlaken TX CRC-32 Generator	**On/Off**	When you turn this option on, the TX Enhanced PCS datapath enables the CRC32 generator function.
Enable Interlaken RX CRC-32 generator error insertion	**On/Off**	When you turn this option on, the error insertion of the interlaken CRC-32 generator is enabled. Error insertion is cycle-accurate. When this feature is enabled, the assertion of `tx_control[8]` or `tx_err_ins` signal causes the CRC calculation during that word is incorrectly inverted, and thus, the CRC created for that metaframe is incorrect.
Enable Interlaken RX CRC-32 checker	**On/Off**	Enables the CRC-32 checker function.
Enable rx_enh_crc32_err port	**On/Off**	When you turn this option on, the Enhanced PCS enables the `rx_enh_crc32_err` port. This signal is asserted to indicate that the CRC checker has found an error in the current metaframe.

Table 2-14: 10GBASE-R BER Checker Parameters

Parameter	Range	Description
Enable rx_enh_highber port (10GBASE-R)	**On/Off**	Enables the `rx_enh_highber` port.
Enable rx_enh_highber_clr_cnt port (10GBASE-R)	**On/Off**	Enables the `rx_enh_highber_clr_cnt` input port.
Enable rx_enh_clr_errblk_count port (10GBASE-R)	**On/Off**	Enables the `rx_enh_clr_errblk_count` input port.

Table 2-15: 64b/66b Encoder and Decoder Parameters

Parameter	Range	Description
Enable TX 64b/66b encoder	**On/Off**	When you turn this option on, the Enhanced PCS enables the TX 64b/66b encoder.
Enable RX 64b/66b decoder	**On/Off**	When you turn this option on, the Enhanced PCS enables the RX 64b/66b decoder.
Enable TX sync header error insertion	**On/Off**	When you turn this option on, the Enhanced PCS supports cycle-accurate error creation to assist in exercising error condition testing on the receiver. When error insertion is enabled and the error flag is set, the encoding sync header for the current word is generated incorrectly. If the correct sync header is 2'b01 (control type), 2'b00 is encoded. If the correct sync header is 2'b10 (data type), 2'b11 is encoded.

Table 2-16: Scrambler and Descrambler Parameters

Parameter	Range	Description
Enable TX scrambler (10GBASE-R/ Interlaken)	**On/Off**	Enables the scrambler function. This option is available for the Interlaken and 10GBASE-R protocols.
TX scrambler seed (10GBASE-R/Interlaken)	User-specified 58-bit value	You must provide a non all-zero seed for the Interlaken protocol. For a multi-lane Interlaken Transceiver Native PHY IP, the first lane scrambler has this seed, and other lanes' scrambler have this seed increased by 1 per lane. The initial seed for 10GBASE-R is 0x03FFFFFFFFFFFFFF. This parameter is required for the 10GBASE-R and Interlaken protocols.

Parameter	Range	Description
Enable RX descrambler (10GBASE-R/ Interlaken)	**On/Off**	Enables the descrambler function. This option is available for the Interlaken and 10GBASE-R protocols.

Table 2-17: Interlaken Disparity Generator and Checker Parameters

Parameter	Range	Description
Enable Interlaken TX disparity generator	**On/Off**	When you turn this option on, the Enhanced PCS enables the disparity generator. This option is available for the Interlaken protocol.
Enable Interlaken RX disparity checker	**On/Off**	When you turn this option on, the Enhanced PCS enables the disparity checker. This option is available for the Interlaken protocol.

Table 2-18: Block Synchronization

Parameter	Range	Description
Enable RX block synchronizer	**On/Off**	When you turn this option on, the Enhanced PCS enables the RX block synchronizer. This option is available for the Interlaken and 10GBASE-R protocols.
Enable rx_enh_ blk_lock port	**On/Off**	Enables the `rx_enh_blk_lock` port.

Table 2-19: Gearbox Parameters

Parameter	Range	Description
Enable TX data bitslip	**On/Off**	When you turn this option on, the TX gearbox operates in bitslip mode.
Enable TX data polarity inversion	**On/Off**	When you turn this option on, the gearbox inverts the polarity of TX data allowing you to correct incorrect placement and routing on the PCB.
Enable RX data bitslip	**On/Off**	When you turn this option on, the Enhanced PCS RX block synchronizer operates in bitslip mode.
Enable RX data polarity inversion	**On/Off**	When you turn this option on, the gearbox inverts the polarity of RX data allowing you to correct incorrect placement and routing on the PCB.
Enable tx_enh_ bitslip port	**On/Off**	Enables the `tx_enh_bitslip` port.
Enable rx_bitslip port	**On/Off**	Enables the `rx_bitslip` port.

Table 2-20: KR-FEC Parameters

Parameter	Range	Description
Enable RX KF-REC error marking	**On/Off**	When you turn this option on, the decoder asserts both sync bits (2'b11) when it detects an uncorrectable error. This feature increases the latency through the KR-FEC decoder.
Enable tx_enh_ frame port	**On/Off**	Enables the `tx_enh_frame` port.
Enable rx_enh_ frame port	**On/Off**	Enables the `rx_enh_frame` port.
Enable rx_enh_ frame_diag_ status port	**On/Off**	Eables the `rx_enh_frame_diag_status` port.

Related Information

- **Arria 10 Enhanced PCS Architecture** on page 5-14
- **Using the "Basic" and "Basic with KR FEC" Configurations of Enhanced PCS** on page 2-225
- **Interlaken** on page 2-62
- **10GBASE-R and 10GBASE-R 1588**
- **10GBASE-KR PHY IP with FEC Option** on page 2-103
- **Enhanced PCS and PMA Ports** on page 2-37

Standard PCS Parameters

This section provides descriptions of the parameters that you can specify to customize the Standard PCS.

The Standard PCS provides Transceiver configuration rules for the following supported protocols:

- Basic
- Basic with Rate Match
- CPRI
- GbE
- GbE 1588

For specific information about configuring the Standard PCS for these protocols, refer to the sections of this user guide that describe support for these protocols.

Table 2-21: Standard PCS Parameters

Note: For detailed descriptions of the optional ports that you can enable or disable, refer to the **Standard PCS and PMA Ports** on page 2-52.

Parameter	Range	Description
Standard PCS/ PMA interface width	**8, 10, 16, 20**	Specifies the data interface width between the Standard PCS and the transceiver PMA.

Parameter	Range	Description
FPGA fabric/ Standard TX PCS interface width	**8, 10, 16, 20, 32, 40**	Shows the FPGA fabric to TX PCS interface width. This value is determined by the current configuration of individual blocks within the Standard TX PCS datapath.
FPGA fabric/ Standard RX PCS interface width	**8, 10, 16, 20, 32, 40**	Shows the FPGA fabric to RX PCS interface width. This value is determined by the current configuration of individual blocks within the Standard RX PCS datapath.
Enable Standard PCS low latency mode	**On/ Off**	Enables the low latency path for the Standard PCS. All individual functional blocks within the Standard PCS are bypassed to provide the lowest latency . You cannot turn on this parameter while using the **Basic/Custom w/Rate Match (Standard PCS)** specified for **Transceiver configuration rules**.

Table 2-22: TX and RX FIFO Parameters

Parameter	Range	Description
TX FIFO mode	**low_latency** **register_fifo**	Specifies the Standard PCS TX FIFO mode. The following 2 modes are available: • **low_latency** : This mode adds 2-3 cycles of latency to the TX datapath. • **register_fifo** : In this mode the FIFO is replaced by registers to reduce the latency through the PCS. Use this mode for protocols that require deterministic latency, such as CPRI.
RX FIFO mode	**low_latency** **register_fifo**	The following 2 modes are available: • **low_latency** : This mode adds 2-3 cycles of latency to the RX datapath. • **register_fifo** : In this mode the FIFO is replaced by registers to reduce the latency through the PCS. Use this mode for protocols that require deterministic latency, such as CPRI.
Enable tx_std_ pcfifo_full port	**On/Off**	Enables the `tx_std_pcfifo_full` port.
Enable tx_std_ pcfifo_empty port	**On/Off**	Enables the `tx_std_pcfifo_empty` port.
Enable rx_std_ pcfifo_full port	**On/Off**	Enables the `rx_std_pcfifo_full` port.
Enable rx_std_ pcfifo_empty port	**On/ Off**	Enables the `rx_std_pcfifo_empty` port.

Table 2-23: Byte Serializer and Deserializer Parameters

Parameter	Range	Description
Enable TX byte serializer	**Disabled** **Serialize x2** **Serialize x4**	Specifies the TX byte serializer mode for the Standard PCS. The transceiver architecture allows the Standard PCS to operate at double or quadruple the data width of the PMA serializer. The byte serializer allows the PCS to run at a lower internal clock frequency to accommodate a wider range of FPGA interface widths. **Serialize x4** is only applicable in PCIe mode.
Enable RX byte deserializer	**Disabled**, **Deserialize x2** **Deserialize x4** (PCIe mode only)	Specifies the mode for the RX byte deserializer in the Standard PCS. The transceiver architecture allows the Standard PCS to operate at double or quadruple the data width of the PMA deserializer. The byte deserializer allows the PCS to run at a lower internal clock frequency to accommodate a wider range of FPGA interface widths.

Table 2-24: 8B/10B Encoder and Decoder Parameters

Parameter	Range	Description
Enable TX 8B/10B encoder	**On/Off**	Enables the `rx_std_rmfifo_full` port.
Enable TX 8B/10B disparity control	**On/Off**	When you turn this option on, the Standard PCS includes disparity control for the 8B/10B encoder. You can force the disparity of the 8B/10B encoder using the `tx_forcedisp` control signal.
Enable RX 8B/10B decoder	**On/Off**	When you turn this option on, the Standard PCS includes the 8B/10B decoder.

Table 2-25: Rate Match FIFO Parameters

Parameter	Range	Description
RX rate match FIFO mode	**Disabled**, **Basic (single width)**, **Basic (double width)**, **GIGE**, **PIPE**, **PIPE 0 ppm**	Specifies the operation of the RX rate match FIFO in the Standard PCS.
RX rate match insert/delete +ve pattern (hex)	User-specified 20 bit pattern	Specifies the +ve (positive) disparity value for the RX rate match FIFO as a hexadecimal string.
RX rate match insert/delete -ve pattern (hex)	User-specified 20 bit pattern	Specifies the -ve (negative) disparity value for the RX rate match FIFO as a hexadecimal string.
Enable rx_std_rmfifo_full port	**On/Off**	Enables the optional `rx_std_rmfifo_full` port.

Parameter	Range	Description
Enable rx_std_rmfifo_empty port	**On/Off**	Enables the `rx_std_rmfifo_empty` port.
PCI Express Gen3 rate match FIFO mode	**Bypass, 0 ppm, 600 ppm**	Specifies the PPM tolerance for the PCI Express Gen3 rate match FIFO.

Table 2-26: Word Aligner and Bitslip Parameters

Parameter	Range	Description
Enable TX bitslip	**On/Off**	When you turn this option on, the PCS includes the bitslip function. The outgoing TX data can be slipped by the number of bits specified by the `tx_std_bitslipboundarysel` control signal.
Enable tx_std_bitslipboundarysel port	**On/Off**	Enables the `tx_std_bitslipboundarysel`.
RX word aligner mode	**bitslip** **manual (PLD controlled)** **synchronous state machine** **deterministic latency**	Specifies the RX word aligner mode for the Standard PCS. The word aligned width depends on the PCS and PMA width, and whether 8B/10B is enabled.
RX word aligner pattern length	**7, 8, 10,16, 20, 32, 40**	Specifies the length of the pattern the word aligner uses for alignment.
RX word aligner pattern (hex)	User-specified	Specifies the word aligner pattern in hex.
Number of word alignment patterns to achieve sync	`0-255`	Specifies the number of valid word alignment patterns that must be received before the word aligner achieves synchronization lock. The default is 3.
Number of invalid words to lose sync	`0-63`	Specifies the number of invalid data codes or disparity errors that must be received before the word aligner loses synchronization. The default is 3.
Number of valid data words to decrement error count	`0-255`	Specifies the number of valid data codes that must be received to decrement the error counter. If the word aligner receives enough valid data codes to decrement the error count to 0, the word aligner returns to synchronization lock.

Parameter	Range	Description
Enable rx_std_wa_patternalign port	**On/Off**	Enables the `rx_std_wa_patternalign` port.
Enable rx_std_wa_a1a2size port	**On/Off**	Enables the `rx_std_wa_a1a2size` port.
Enable rx_std_bitslipboundarysel port	**On/Off**	Enables the `rx_std_bitslipboundarysel` port.
Enable rx_bitslip port	**On/Off**	Enables the `rx_bitslip` port.

Table 2-27: Bit Reversal and Polarity Inversion

Parameter	Range	Description
Enable TX bit reversal	**On/Off**	When you turn this option on, the 8B/10B Encoder reverses TX parallel data before transmitting it to the PMA for serialization. The transmitted TX data bit order is reversed to MSB to LSB rather than the normal LSB to MSB. This is a static setting and can only be changed dynamically through dynamic reconfiguration.
Enable TX byte reversal	**On/Off**	When you turn this option on, the 8B/10B Encoder reverses the byte order before transmitting data. This function allows you to reverse the order of bytes that were erroneously swapped. The PCS can swap the ordering of both 8-bit and 10-bit words. When the PCS to PMA interface width is 16 or 20 bits, the PCS can swap the ordering of the individual 8-bit or 10-bit words. This option is not valid under some **Transceiver configuration rules**.
Enable TX polarity inversion	**On/Off**	When you turn this option on, the `tx_std_polinv` port controls polarity inversion of TX parallel data to the PMA. When you turn on this parameter, you also need to turn on **Enable tx_polinv port**.
Enable tx_polinv port	**On/Off**	When you turn this option on, the `tx_polinv` input control port is enabled. You can use this control port to swap the positive and negative signals of a serial differential link if they were erroneously swapped during board layout.
Enable RX bit reversal	**On/Off**	When you turn this option on, the word aligner reverses RX parallel data. The received RX data bit order is reversed to MSB to LSB rather than the normal LSB to MSB. This is a static setting and can only be changed dynamically through dynamic reconfiguration. When you enable **Enable RX bit reversal**, you must also enable **Enable rx_std_bitrev_ena port**.

Parameter	Range	Description
Enable rx_std_bitrev_ena port	**On/Off**	When you turn this option on, asserting the `rx_std_bitrev_ena` control port causes the RX data order to be reversed from the normal order, LSB to MSB, to the opposite, MSB to LSB.
Enable RX byte reversal	**On/Off**	When you turn this option on, the word aligner reverses the byte order before storing the data in the RX FIFO. This function allows you to reverse the order of bytes that were erroneously swapped. The PCS can swap the ordering of both 8 and10 bit words. When the PCS / PMA interface width is 16 or 20 bits the PCS can swap the ordering of the individual 8- or 10-bit words. This option is not valid under some **Transceiver configuration rules**. When you enable **Enable RX byte reversal**, you must also enable **Enable rx_std_byterev_ena port**.
Enable rx_std_byterev_ena port	**On/Off**	When you turn this option on, asserting `rx_std_byterev_ena` input control port causes swaps the order of the individual 8- or 10-bit words received from the PMA.
Enable RX polarity inversion	**On/Off**	When this option is on, the `rx_std_polinv` port controls polarity inversion of RX parallel data. When you turn on this parameter, you also need to enable **Enable rx_polinv port**.
Enable rx_polinv port	**On/Off**	When you turn this option on, the `rx_polinv` input is enabled. You can use this control port to swap the positive and negative signals of a serial differential link if they were erroneously swapped during board layout.
Enable rx_std_signaldetect port	**On/Off**	When you turn this option on, the optional `rx_std_signaldetect` output port is enabled. This signal is required for the PCI Express protocol. If enabled, the signal threshold detection circuitry senses whether the signal level present at the RX input buffer is above the signal detect threshold voltage that you specified. You can specify the signal detect threshold using a Quartus II QSF assignments.

Table 2-28: PCIe Parameters

Parameter	Range	Description
Enable PCIe dynamic datarate switch ports	**On/Off**	When you turn this option on, the `pipe_rate` and `pipe_sw_done` ports are enabled. You should connect these ports to the PLL IP instance in multi-lane PCIe Gen2 and Gen3 configurations. These ports are only available for multi-lane bonded configurations.
Enable PCIe pipe_hclk_in and pipe_hclk_out ports	**On/Off**	When you turn this option on, enables the `pipe_hclk_in` and `pipe_hclk_out`. These ports must be connected to the PLL IP instance for the PCI Express core.

Parameter	Range	Description
Enable PCIe Gen3 analog control ports	**On/Off**	When you turn this option on, enables the `pipe_g3_txdeemph` and `pipe_g3_rxpresenthint` ports. You can use these ports to for equalization for Gen3 configurations.
Enable PCIe electrical idle control and status ports	**On/Off**	When you turn this option on, enables the `pipe_rx_eidleinfersel` and `pipe_rx_elecidle` ports. These ports are used for PCI Express configurations.
Enable PCIe pipe_rx_polarity port	**On/Off**	When you turn this option on, enables the `pipe_rx_polarity` input control port. You can use this to control channel signal polarity for PCI Express configurations. When the Standard PCS is configured for PCIe, the assertion of this signal causes the RX bit polarity to be inverted. For other **Transceiver configuration rules** the optional `rx_polinv` port inverts the polarity of the RX bit stream.

Related Information

Standard PCS and PMA Ports on page 2-52

Dynamic Reconfiguration Parameters

Dynamic reconfiguration allows you to change the behavior of the transceiver channels and PLLs without powering down the device. Each transceiver channel and PLL includes an Avalon-MM slave interface for reconfiguration. This interface provides direct access to the programmable address space of each channel and PLL. Because each channel and PLL includes a dedicated Avalon-MM slave interface, you can dynamically modify channels either concurrently or sequentially. If your system does not require concurrent reconfiguration, you can parameterize the Transceiver Native PHY IP to share a single reconfiguration interface.

You can use dynamic reconfiguration to change many functions and features of the transceiver channels and PLLs. For example, you can change the reference clock input to the TX PLL. You can also change between the Standard and Enhanced datapaths.

Table 2-29: Dynamic Reconfiguration

Parameter	Value	Description
Enable dynamic reconfiguration	**On/Off**	When you turn this option on, the dynamic reconfiguration interface is enabled.
Share reconfiguration interface	**On/Off**	When you turn this option on, the Transceiver Native PHY IP presents a single Avalon-MM slave interface for dynamic reconfiguration for all channels. In this configuration, the upper [*n*:10] address bits of the reconfiguration address bus specify the channel. The channel numbers are binary encoded. Address bits [9:0] provide the register offset address within the reconfiguration space for a channel.

Parameter	Value	Description
Enable embedded JTAG Avalon-MM master	**On/Off**	When you turn this option on, the Transceiver Native PHY IP includes an embedded JTAG Avalon-MM master that connects to the Avalon-MM slave interface for dynamic reconfiguration. The JTAG Avalon-MM master can access the reconfiguration space of the transceiver. It can perform certain test and debug functions via JTAG using the System Console. This option requires you to enable the **Share reconfiguration interface** option.

Table 2-30: Configuration Files

Parameter	Value	Description
Configuration file prefix	*<prefix>*	When you turn this option on, it specifies the file prefix to use for generated configuration files. Each variant of the Transceiver Native PHY IP should use a unique prefix for configuration files.
Generate SystemVerilog package file	**On/Off**	When you turn this option on, the Transceiver Native PHY IP generates a SystemVerilog package file, **_reconifg_parameters.sv**, containing parameters defined with the attribute values required for reconfiguration.
Generate C header file	**On/Off**	When you turn this option on, the Transceiver Native PHY IP generates a C header file, **_reconifg_parameters.h**, containing macros defined with the attribute values required for reconfiguration.
Generate MIF (Memory Initialization File)	**On/Off**	When you turn this option on, the Transceiver Native PHY IP generates a MIF,**_reconifg_parameters.mif**, containing the attribute values required for reconfiguration in a data format.

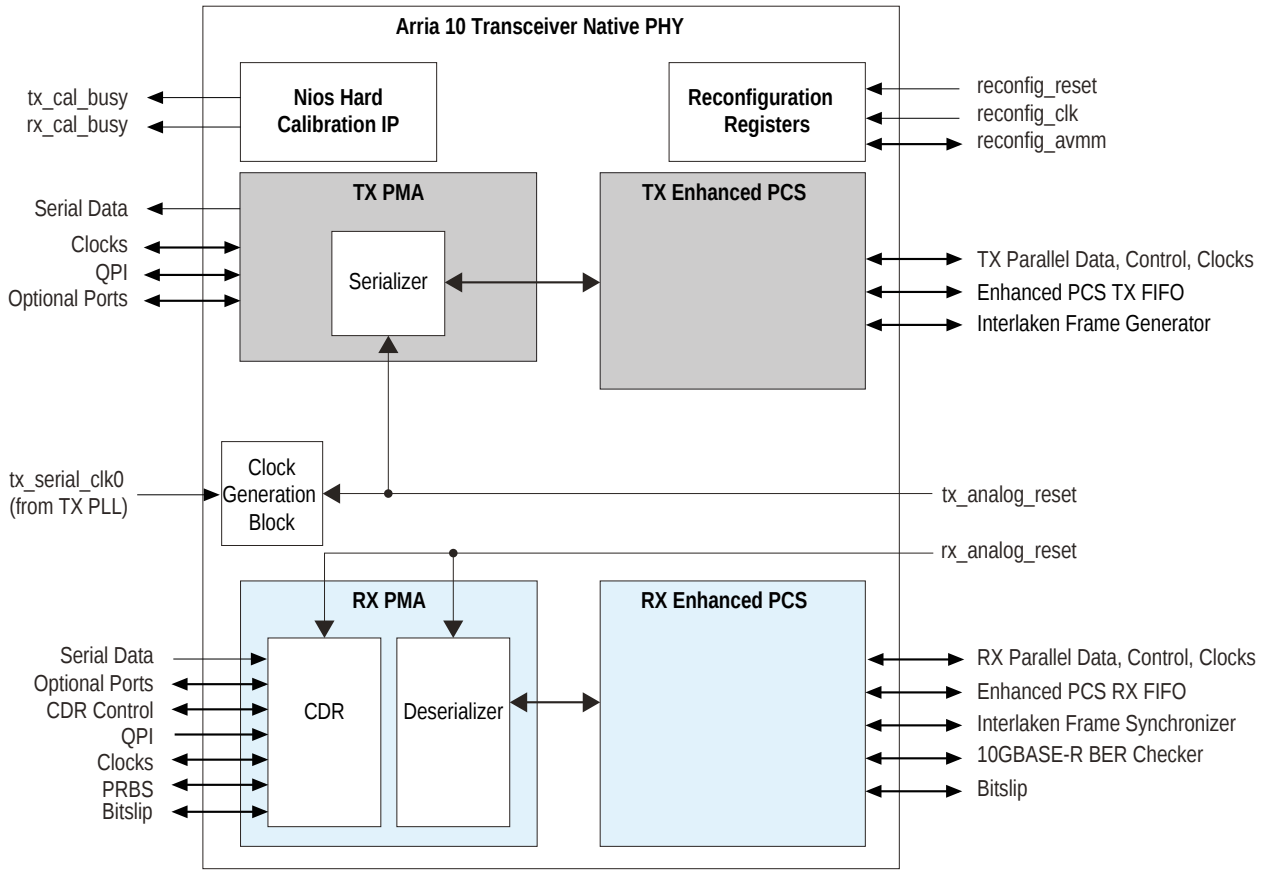
Table 2-31: Generation Options

Parameter	Value	Description
Generate parameter documentation file	**On/Off**	When you turn this option on, generation produces a Comma-Separated Value File (**.csv**) with descriptions of the Transceiver Native PHY IP parameters.

Enhanced PCS and PMA Ports

Figure 2-7: Enhanced PCS Interfaces

If you enable both the Enhanced PCS and Standard PCS your top-level HDL file includes all the ports for both datapaths. The labeled inputs and outputs to the PMA and PCS modules represent buses, not individual signals.



In the following tables, the variables represent these parameters:

- <*n*>—The number of lanes
- <*d*>—The serialization factor
- <*s*>— The symbol size
- <*p*>—The number of PLLs

Table 2-32: Enhanced TX PCS: Parallel Data, Control, and Clocks

Name	Direction	Clock Domain	Description
`tx_parallel_data[<n>128-1:0]`	Input	`tx_coreclkin`	TX parallel data inputs from the FPGA fabric to the TX PCS. If you select, **Enable simplified interface** in the Transceiver Native PHY IP GUI, `tx_parallel_data` includes only the bits required for the configuration you specify.

Name	Direction	Clock Domain	Description
			When FPGA fabric/PCS interface data width is 64 bits, you must ground data[127:64]. The following bits are active for narrower interfaces: • 32-bit FPGA fabric IF width: tx_parallel_data[31:0]. Ground [63:32]. • 40-bit FPGA fabric IF width: tx_parallel_data[39:0]. Ground [63:40]. • 64-bit FPGA fabric IF width: tx_parallel_data[63:0]. When the FPGA fabric/PCS interface data width is 128 bits (double-width mode), the following bits are active for narrower double-width configurations: • 32-bit FPGA fabric IF width: data[95:64]. Ground[127:96]. • 40-bit FPGA fabric IF width: data[103:64]. Ground [127:104]. • 64-bit FPGA fabric IF width: data[127:64]. **Note:** You cannot select the **Enable simplified interface** if you plan to dynamically reconfigure between multiple protocols.
`unused_tx_parallel_data`	Input	`tx_clkout`	This signal specifies the unused data when you turn on **Enable simplified data interface**. Connect all of these bits to 0.
`tx_control[<n><3>-1:0]` or `tx_control[<n><18>-1:0]`	Input	`tx_coreclkin`	Indicates whether the `tx_parallel_data` bus is control or data. If you select, **Enable simplified interface** in the Transceiver Native PHY IP GUI, `tx_control` is 3 bits. If you do not select **Enable simplified interface**, `tx_control` is 18 bits. The following encodings are defined.

Name	Direction	Clock Domain	Description

			• Interlaken: With **Enable simplified interface** on:
			  • [2]: Inversion control. Unused. Ground. Instead of controlled by this bit, the Interlaken running disparity is internally maintained by Enhanced PCS built-in disparity generator block.
			  • [1]: Sync Header (high indicates a control word, low indicates a data word.) Bit[1] is mutually exclusive with bit[0].
			  • [0]: Sync Header (high indicates a control word, low indicates a data word.) Bit[0] is mutually exclusive with bit[1].
			• Interlaken: With **Enable simplified interface** off:
			  • [17:9]: Unused. Ground.
			  • [8]: You can use this bit to insert sync header error or CRC32 errors. Functions like `tx_err_ins`. Refer to the description of `tx_err_ins` for detailed information.
			  • [7:3]: Unused. Ground.
			  • [2]: Inversion control. Unused. Ground. Instead of controlled by this bit, the Interlaken running disparity is internally maintained by Enhanced PCS built-in disparity generator block .
			  • [1]: Sync Header (high indicates a control word, low indicates a data word.) Bit[1] is mutually exclusive with bit[0].
			  • [0]: Sync Header (high indicates a control word, low indicates a data word.) Bit[0] is mutually exclusive with bit[1].

Name	Direction	Clock Domain	Description

			• 10GBASE-R:
			  • [8]: Active-high synchronous error insertion control bit
			  • [7]: XGMII control signal for `tx_parallel_data[63:56]`.
			  • [6]: XGMII control signal for `tx_parallel_data[55:48]`.
			  • [5]: XGMII control signal for`tx_parallel_data[47:40]`.
			  • [4]: XGMII control signal for `tx_parallel_data[39:32]`.
			  • [3]: XGMII control signal for `tx_parallel_data[31:24]`.
			  • [2]: XGMII control signal for`tx_parallel_data[23:16]`.
			  • [1]: XGMII control signal for `tx_parallel_data[15:8]`.
			  • [0]: XGMII control signal for `tx_parallel_data[7:0]`.
			• 10GBASE-R KR FEC and Basic KRFEC:
			  • [9]: Active-high status signal that indicates when KRFEC Block Lock is achieved.
			  • [8]: Active-high status signal that indicates the beginning of a received KRFEC frame boundary.
			• Basic mode, 64-bit data, including sync header:
			  • [8:2]: Unused. Ground.
			  • [1]: Sync Header. 1 indicates a control word.
			  • [0]: Sync Header. 1 indicates a data word.
			• Basic, 64-bit data, excluding sync header:
			  • [8:0]: Unused. Ground.
			• Basic mode, 128-bit data, including sync header:
			  • [17:11]: Unused. Ground.
			  • [10]: Sync Header, 1 indicates a control word).
			  • [9]: Sync Header, 1 indicates a data word).
			• Basic mode, 128-bit data, excluding synchronization header:
			  • [17:9]: Unused. Ground.

`unused_tx_control[<n> <15>-1:0]`	Input	`tx_coreclkin`	Port is enabled when you enable **Enable simplified interface**. Connect all of these bits to 0.

Name	Direction	Clock Domain	Description
tx_err_ins	Input	tx_coreclkin	For the Interlaken protocol, you can use this bit to insert sync header and CRC32 errors if you have turned on **Enable simplified interface**. When asserted, the sync header for that cycle word is replaced with a corrupted one. A CRC32 error is also inserted if **Enable Interlaken TX CRC-32 generator error insertion** is turned on. The corrupted sync header is 2'b00 for a control word, and 2'b11 for a data word. For CRC32 error insertion, the word used for CRC calculation for that cycle is incorrectly inverted, causing an incorrect CRC32 in the Diagnostic Word of the Metaframe. Note that a sync header error and a CRC32 error cannot be created for the Framing Control Words because the Frame Control Words are created in the frame generator embedded in TX PCS. Both the sync header error and the CRC32 errors are inserted if the CRC-32 error insertion feature is enabled in the Transceiver Native PHY IP GUI.
tx_coreclkin	Input	Clock	The FPGA fabric clock. Drives the write side of the TX FIFO. For the Interlaken protocol, the frequency of this clock could be from datarate/67 to datarate/32. However, if the frequency is too much lower than tx_clkout, the TX FIFO flags may be unable to update in time and cause data corruption.
tx_clkout	Output	Clock	The parallel clock generated by the transceiver TX PMA. This clock times the blocks of the TX Enhanced PCS. The frequency of this clock is equal to datarate divided by PCS/PMA interface width.
tx_serial_data	Output	N/A	Serial data output from the TX PMA .
tx_serial_clk0	Input	Clock	Serial Clock input from the TX PLL. For non-bonded channels only. The frequency of this clock depends on the data rate and clock division factor. tx_serial_clk0 is for non bonded channels only. For bonded channels use the tx_bonding_clocks clock TX input.
tx_bonding_clocks[<n><6>-1:0]	Input	Clock	6-bit bus which carries the low speed parallel clock per channel. Outputs from the Master CGB. Use for bonded channels only.
Optional Ports			

Name	Direction	Clock Domain	Description
tx_serial_clk1 tx_serial_clk2 tx_serial_clk3 tx_serial_clk4	Inputs	Clocks	Serial Clock inputs from the TX PLL. The frequency of this clock depends on the data rate and clock division factor. When you specify more than 1 TX PLLs, these additional clock ports are enabled.
tx_pma_clkout	Output	Clock	This is the parallel clock from the TX PMA. It is available when you turn on **Enable tx_pma_clkout port** in the Transceiver Native PHY IP GUI.
tx_pma_div_clkout	Output	Clock	This is a divided version of the recovered parallel clock. It is available when you turn on **Enable tx_pma_div_clkout port** in the Transceiver Native PHY IP GUI. You can divide the parallel clock by 1 or 2. You can divide the serial clock by 33, 40, or 66.
tx_pma_elecidle[<n>-1:0]	Input	Asynchronous	When asserted this signal forces the transmitter to electrical idle. This port has no effect when you configure the transceiver for the PCI Express protocol.
tx_pma_qpipullup[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_qpipullup port (QPI)** in the Transceiver Native PHY IP GUI. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_qpipulldn[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_qpipulldn port (QPI)** in the Transceiver Native PHY IP GUI. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_txdetectrx[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_txdetectrx port (QPI)** in the Transceiver Native PHY IP GUI. When asserted, the receiver detect block in TX PMA detects the presence of a transmitter at the other end of the channel. After receiving tx_pma_txdetectrx request the receiver detect block initiates the detection process. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_rxfound[<n>-1:0]	Output	Asynchronous	This port is available if you turn on **Enable tx_rxfound port (QPI)** in the Transceiver Native PHY IP GUI. When asserted, indicates that the receiver detect block in TX PMA has detected a transmitter at the other end of the channel. It is only used for Quick Path Interconnect (QPI) applications.

Table 2-33: Enhanced RX PCS: Parallel Data, Control, and Clocks

Name	Direction	Clock Domain	Description
rx_parallel_data[<n>128-1:0]	Output	rx_coreclkin	RX parallel data from the RX PCS to the FPGA fabric. If you select, **Enable simplified interface** in the Transceiver Native PHY IP GUI, rx_parallel_data includes only the bits

Name	Direction	Clock Domain	Description
			required for the configuration you specify. Otherwise, this interface is 128 bits wide. When FPGA fabric/PCS interface width is 64 bits (single-width mode), you must ground data[127:64]. The following bits are active for narrower interfaces: • 32-bit FPGA fabric IF width: data[31:0]. Unused. • 40-bit FPGA fabric IF width: data[39:0]. Unused. • 50-bit FPGA fabric IF width: data[49:0], remaining [63:50] unused • 64-bit FPGA fabric IF width: data[63:0]. When the FPGA fabric/PCS interface width is 128 bits, the following bits are active: • 32-bit FPGA fabric IF width: data[95:64]. Unused. • 40-bit FPGA fabric IF width: data[103:64]. Unused. • 50-bit FPGA fabric IF width: data[113:64], remaining [127:114] unused • 64-bit FPGA fabric IF width: data[127:64].
`unused_rx_parallel_data`	Output	`rx_clkout`	This signal specifies the unused data when you turn on **Enable simplified data interface**.

Name	Direction	Clock Domain	Description

rx_control[<n><20>-1:0]	Output	rx_coreclkin	Indicates whether the rx_parallel_data bus is control or data. Defined for the following configurations:
			• Interlaken: With **Enable simplified interface** on:
			  • [19:10]: Unused.
			  • [9]: When 1'b1, indicates that block lock and frame lock have been achieved.
			  • [8]: When 1'b1, indicates a synchronization header error, Metaframe error, or CRC32 error.
			  • [7]: When 1'b1, indicates the Diagnostic Word location within a Metaframe.
			  • [6]: When 1'b1, indicates the SKIP Word location with a Metaframe.
			  • [5]: When 1'b1, indicates the Scrambler State Word location in a Metaframe.
			  • [4]: When 1b'1, indicates the Synchronization Word location in a Metaframe.
			  • [3]: When 1b'1, indicates the Payload Word location in a Metaframe.
			  • [2]: Inversion bit. In the current implementation, this bit is always 0.
			  • [1]: Synchronization header (1 indicates a control word).
			  • [0]: Synchronization header (1 indicates a data word)
			• Interlaken: With **Enable simplified interface** off:
			  • [19:10]: Same as unused_rx_control[9:0]. Can be left floating.
			  • [9:0]: Same as above when you turn on **Enable simplified interface**. Can be left floating.

Name	Direction	Clock Domain	Description

			• 10GBASE-R:
			  • [9]: Active-high status signal that indicates when Block Lock is achieved.
			  • [8]: Active-high status signal that indicates a syn header error.
			  • [7]: XGMII control signal for `rx_parallel_data[63:56]`.
			  • [6]: XGMII control signal for `rx_parallel_data[55:48]`.
			  • [5]: XGMII control signal for `rx_parallel_data[47:40]`.
			  • [4]: XGMII control signal for `rx_parallel_data[39:32]`.
			  • [3]: XGMII control signal for `rx_parallel_data[31:24]`.
			  • [2]: XGMII control signal for `rx_parallel_data[23:16]`.
			  • [1]: XGMII control signal for `rx_parallel_data[15:8]`.
			  • [0]: XGMII control signal for `rx_parallel_data[7:0]`.
			• Basic mode, 64-bit data, including synchronization header:
			  • [9]: Active-high status signal indicating when Block Lock is achieved.
			  • [8]: Active-high status signal that indicates a sync header error.
			  • [7:2]: Unused.
			  • [1]: Sync Header, 1 indicates a control word.
			  • [0]: Sync Header, 1 indicates a data word.
			• Basic, 64-bit data, excluding synchronization header:
			  • [9:0]: Unused.
			• Basic mode, 128-bit data, including synchronization header:
			  • [19]: Active-high status signal indicating when Block Lock is achieved.
			  • [18]: Active-high status signal that indicates a sync header error.
			  • [17:12]: Unused.
			  • [11]: Sync Header, 1 indicates a control word.
			  • [10]: Sync Header, 1 indicates a data word.
			• Basic mode, 128-bit data, excluding synchronization header:
			  • [19:10]: Unused.

Name	Direction	Clock Domain	Description
unused_rx_control[<*n*>10-1:0]	Output	rx_coreclkin	These signals only exist when you turn on **Enable simplified interface**. These outputs can be left floating.
rx_coreclkin	Input	Clock	The FPGA fabric clock. Drives the read side of the RX FIFO. For Interlaken protocol, the frequency of this clock could be from datarate/67 to datarate/32. If the frequency is too much higher than the frequency of rx_clkout, the RX FIFO flags might be unable to update in time and cause data corruption.
rx_clkout	Output	Clock	The low speed parallel clock generated by the transceiver RX PMA. This clock times the blocks in the RX Enhanced PCS. The frequency of this clock is equal to data rate divided by PCS/PMA interface width.
rx_serial_data	Input	N/A	Serial data input to the RX PMA .
rx_cdr_refclk0	Input	Clock	Reference clock input to the RX clock data recovery (CDR) circuitry.
Optional Ports			
rx_cdr_refclk1 - rx_cdr_refclk1	Inputs	Clock	Reference clock input to the RX clock data recovery (CDR) circuitry.
rx_pma_qpipullup[<n>-1:0]	Input	Asynchronous	This port is only used for Quick Path Interconnect (QPI) applications.
rx_is_lockedtoref[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the CDR PLL is locked to the incoming data, rx_serial_data.
rx_is_lockedtoref[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the CDR PLL is locked to the incoming data, rx_serial_data.
rx_set_lockedto-data[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_set_lockedtoref[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_seriallp-bken[<n>-1:0]	Input	Asynchronous	When asserted, enables the TX to RX serial loopback path in the transceiver.
rx_prbs_done[<n>-1:0]	Output	rx_coreclkin	When asserted, indicates the verifier has aligned and captured consecutive PRBS patterns and the first pass through a polynomial is complete. The generator has restarted the polynomial.
rx_prbs_err[<n>-1:0]	Output	rx_coreclkin	When asserted, indicates an error only after the rx_prbs_done signal has been asserted. This signal pulses for every error that occurs. Errors can only occur once per word.

Name	Direction	Clock Domain	Description
rx_prbs_err_clr[<n>-1:0]	Input	rx_coreclkin	When asserted, clears the PRBS pattern and deasserts the rx_prbs_done signal.
rx_pma_clkout	Output	Clock	This is the recovered parallel clock from the RX CDR circuitry.
rx_pma_div_clkout	Output	Clock	This is a divided version the recovered RX parallel clock. For the parallel clock, you can divide rx_pma_clkout by 1 or 2. For the serial clock, you can divide by 33, 40, 50, or 66. This clock drives rx_coreclkin for different gear box ratios.
rx_pma_clkslip	Input	Clock	A rising edge on this signal causes the RX deserializer to slip the serial data by one clock cycle (2 UI).

Table 2-34: Enhanced PCS TX FIFO

Name	Direction	Clock Domain	Description
tx_enh_data_valid[<n> -1:0]	Input	tx_coreclkin	Assertion of this signal indicates that the TX data is valid. Connect this signal to 1'b1 for 10GBASE-R without 1588. For Enhanced Basic and 10GBASE-R with 1588, you must control this signal based on the gearbox ratio. For Interlaken, you need to control this port based on TX FIFO flags so that the FIFO won't underflow or overflow.
tx_enh_fifo_full[<n>-1:0]	Output	tx_coreclkin	Active high. Assertion of this signal indicates the TX FIFO is full.
tx_enh_fifo_pfull[<n>-1:0]	Output	Synchronous to tx_coreclkin	Active high. This signal indicates when the TX FIFO reaches its partially full threshold.
tx_enh_fifo_empty[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the TX FIFO is empty.
tx_enh_fifo_pempty[<n>-1:0]	Output	Asynchronous	Active high. When asserted, indicates that the TX FIFO has reached its specified partially empty threshold. When you turn this option on, the Enhanced PCS enables the tx_enh_fifo_pempty port, which is asynchronous.
tx_enh_fifo_cnt[<n>-1:0]	Output	tx_clkout	Indicates the current level of the TX FIFO.

Table 2-35: Enhanced PCS RX FIFO

Name	Direction	Clock Domain	Description
rx_enh_fifo_rd_en[<n>-1:0]	Input	rx_coreclkin	For Interlaken only, when this signal is asserted, a word is read form the RX FIFO. You need to control this signal based on RX FIFO flags so that the FIFO won't underflow or overflow.
rx_enh_data_valid[<n>-1:0]	Output	rx_coreclkin	When asserted, indicates that rx_parallel_data is valid. For basic mode, the rx_enh_data_valid signal toggles, indicating valid RX data when the RX FIFO is in **Phase compensation** or **Register** mode. This option is available when you select the following parameters: • Enhanced PCS **Transceiver configuration rules** specifies Interlaken • Enhanced PCS **Transceiver configuration rules** specifies Basic and RX FIFO mode is **Phase compensation** • Enhanced PCS **Transceiver configuration rules** specifies Basic and RX FIFO mode is **Register**
rx_enh_fifo_full[<n>-1:0]	Output	Asynchronous	Active high. When asserted, indicates that the RX FIFO is full.
rx_enh_fifo_pfull[<n>-1:0]	Output	Asynchronous	Active high. When asserted, indicates that the RX FIFO has reached its specified partially full threshold.
rx_enh_fifo_empty[<n>-1:0]	Output	Synchronous to rx_coreclkin	Active high. When asserted, indicates that the RX FIFO is empty.
rx_enh_fifo_pempty[<n>-1:0]	Output	Synchronous to rx_coreclkin.	Active high. When asserted, indicates that the RX FIFO has reached its specified partially empty threshold.
rx_enh_fifo_del[<n>-1:0]	Output	Asynchronous	When asserted, indicates that a word has been deleted from the RX FIFO. This signal is used for the 10GBASE-R protocol.
rx_enh_fifo_insert[<n>-1:0]	Output	Synchronous to rx_coreclkin	When asserted, indicates that a word has been inserted into the RX FIFO. This signal is used for the 10GBASE-R protocol.
rx_enh_fifo_cnt[<n> 5-1:0]	Output	rx_coreclkin	Indicates the current level of the RX FIFO.

Name	Direction	Clock Domain	Description
rx_enh_fifo_align_val[<n>-1:0]	Output	Synchronous to rx_clkoutkin	When asserted, indicates that the word alignment pattern has been found. This signal is only valid for the Interlaken protocol.
rx_enh_fifo_align_clr[<n>-1:0]	Input	rx_clkoutkin	When asserted, the FIFO resets and begins searching for a new alignment pattern. This signal is only valid for the Interlaken protocol. Assert this signal for at least 4 cycles.

Table 2-36: Interlaken Frame Generator, Synchronizer, and CRC32

Name	Direction	Clock Domain	Description
tx_enh_frame[<n>-1:0]	Output	Asynchronous	Asserted for 2 or 3 parallel clock cycles to indicate the beginning of a new Metaframe.
tx_enh_frame_burst_en[<n>-1:0]	Input	tx_clkout	If **Enable frame burst** is enabled, this port controls frame generator data reads from the TX FIFO to the frame generator. It is latched once at the beginning of each Metaframe. If the value of tx_enh_frame_burst_en is 0, the frame generator does not read data from the TX FIFO for current Metaframe. Instead, the frame generator inserts SKIP words as the payload of Metaframe. When tx_enh_frame_burst_en is 1, the frame generator reads data from the TX FIFO for the current Metaframe. This port must be held constant for 5 clock cycles before and after the tx_enh_frame pulse.
tx_enh_frame_diag_status[<n> 2-1:0]	Input	tx_clkout	Drives the lane status message contained in the Framing Layer Diagnostic Word (bits[33:32]) . This message is inserted into the next Diagnostic Word generated by the Frame Generator Block. This bus must be held constant for 5 clock cycles before and after the tx_enh_frame pulse. The following encodings are defined: • Bit[1]: When 1'b1, indicates the lane is operational. When 1'b0, indicates the lane is not operational. • Bit[0]: When 1'b1, indicates the link is operational. When 1'b0, indicates the link is not operational.
rx_enh_frame[<n>-1:0]	Output	Asynchronous	When asserted, indicates the beginning of a new received Metaframe.
rx_enh_frame_lock[<n>-1:0]	Output	Asynchronous	When asserted, indicates the Frame Synchronizer state machine has achieved Metaframe delineation.

Name	Direction	Clock Domain	Description
rx_enh_frame_diag_status[2 <n>-1:0]	Output	Asynchronous	Drives the lane status message contained in the Framing Layer Diagnostic Word (bits[33:32]) . This signal is latched when a valid Diagnostic Word is received in the end of the Metaframe while the frame is locked. The following encodings are defined: • Bit[1]: When 1b'1, indicates the lane is operational. When 0b'0, indicates the lane is not operational. • Bit[0]: When 1b'1, indicates the link is operational. When 0b'0, indicates the link is not operational.
rx_enh_crc32_err[<n>-1:0]	Output	Asynchronous	When asserted, indicates a CRC error in the current Metaframe. Asserted at the end of current Metaframe. This signal is pulse stretched for 2 or 3 cycles.

Table 2-37: 10GBASE-R BER Checker

Name	Direction	Clock Domain	Description
rx_enh_highber[<n>-1:0]	Output	Asynchronous	Active high. When asserted, indicates a bit error rate that is greater than 10^{-4}. For the 10GBASE-R protocol, this BER rate occurs when there are at least 16 errors within 125 µs.
rx_enh_highber_clr_cnt[<n>-1:0]	Input	Asynchronous. Generated on rx_clkout	When asserted, clears the internal counter that indicates the number of times the BER state machine has entered the BER_BAD_SH state.
rx_enh_clr_errblk_count[<n>-1:0] (10GBASE-R and FEC)	Input	Asynchronous. Generated on rx_clkout	For 10GBASE-R, enables the optional rx_enh_clr_errblk_count input port. When asserted the error block counter resets to 0. Assertion of this signal clears the internal counter that counts the number of times the RX state machine has entered the RX_E state. In modes where the FEC block is enabled, the assertion of this signal resets the status counters within the RX FEC block.

Table 2-38: Block Synchronizer

Name	Direction	Clock Domain	Description
rx_enh_blk_lock<n>-1:0]	Output	Asynchronous	Active high. When asserted, indicates that block synchronizer has achieved block delineation. This signal is used for 10GBASE-R and Interlaken.

Table 2-39: Bitslip

Name	Direction	Clock Domain	Description
`rx_bitslip[<n>-1:0]`	Input	Asynchronous. `rx_clkout`	The `rx_parallel_data` slips 1 bit for every positive edge of the `rx_bitslip` input. The maximum shift is < *pcswidth -1*> bits, so that if the PCS is 64 bits wide, you can shift 0-63 bits.
`tx_enh_bitslip[<n>-1:0]`	Input	Asynchronous. `rx_clkout`	The value of this signal controls the number of cbit locations to slip the `tx_parallel_data` before passing to the PMA.

Related Information

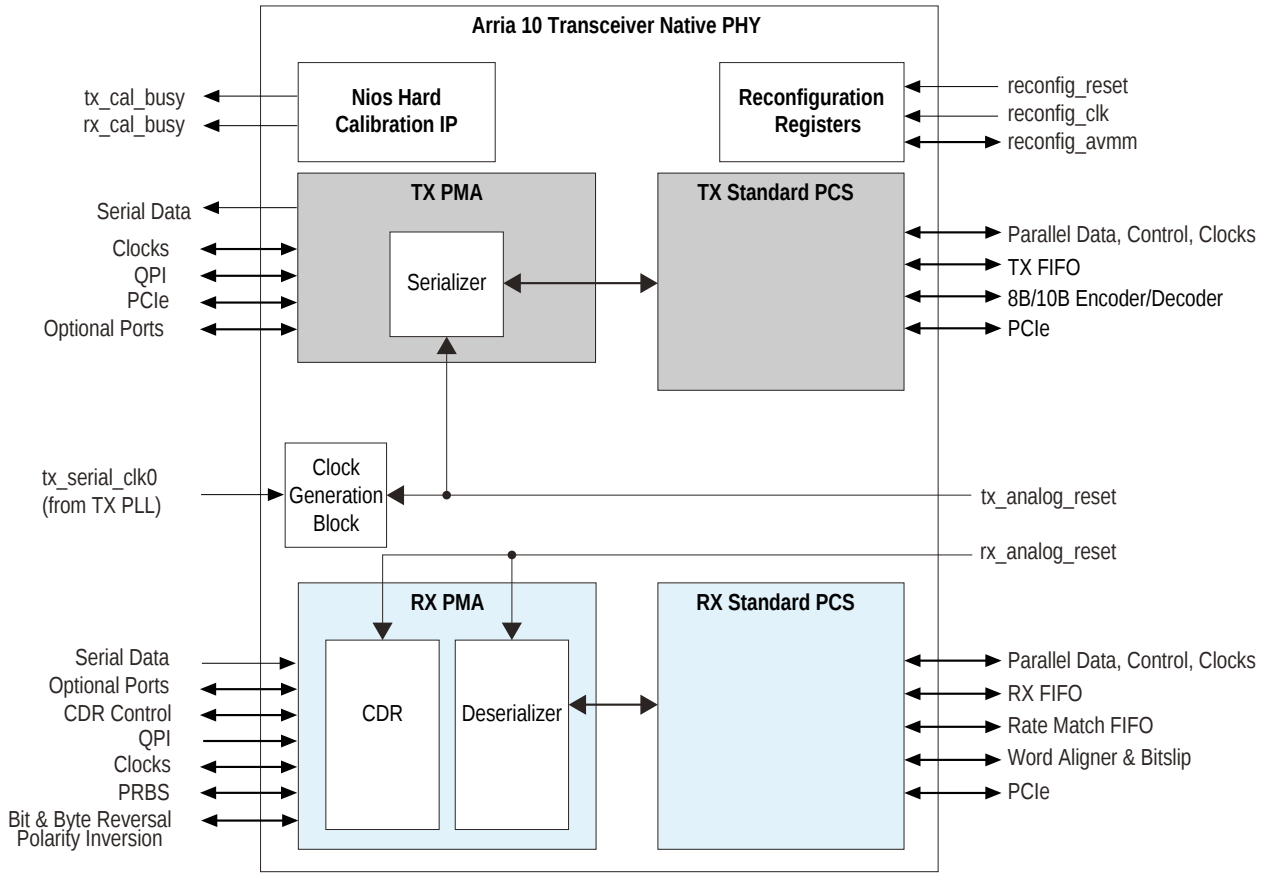
- **ATX PLL IP** on page 3-6
- **CMU PLL IP** on page 3-21
- **fPLL IP** on page 3-13
- **Ports and Parameters** on page 6-1

- **Transceiver PHY Reset Controller Interfaces** on page 4-12

This section describes the top-level signals for the Transceiver PHY Reset Controller IP core.

Standard PCS and PMA Ports

The following figure illustrates the transceiver channel using the Standard PCS. If you enable both the Standard PCS and Enhanced PCS your top-level HDL file includes all the ports for both datapaths.



In the following tables, the variables represent these parameters:

- <*n*>—The number of lanes
- <*w*>—The width of the interface
- <*d*>—The serialization factor
- <*s*>— The symbol size
- <*p*>—The number of PLLs

Table 2-40: TX Standard PCS: Data, Control, and Clocks

Name	Direction	Clock Domain	Description
`tx_parallel_data[<n>128-1:0]`	Input	`tx_clkout`	TX parallel data input from the FPGA fabric to the TX PCS. For each 128-bit word, the data input bits correspond to `tx_parallel_data[7:0]`.

Name	Direction	Clock Domain	Description
unused_tx_parallel_data	Input	tx_clkout	This signal specifies the unused data when you turn on **Enable simplified data interface**. Connect all these bits to 0.
tx_datak[<n><d>/<s>-1:0]	Input	tx_clkout	When 1, indicates that the 8B/10B encoded word of tx_parallel_data is data. When 0, indicates that the 8B/10B encoded word of tx_parallel_data is control. tx_datak is a part of tx_parallel_data. For each 128-bit word, tx_datak corresponds to tx_parallel_data[8].
tx_coreclkin	Input	Clock	The FPGA fabric clock. This clock drives the write port of the TX FIFO.
tx_clkout	Output	Clock	The low speed parallel clock generated by the Transceiver TX PMA. This clock times tx_parallel_data from the FPGA fabric to the TX PCS.

Table 2-41: TX Standard PMA: Data and Optional PMA Ports

Name	Direction	Clock Domain	Description
tx_serial_data[<n>-1:0]	Input	N/A	Serial data output of the TX PMA.
Optional Ports			
tx_serial_clk1 tx_serial_clk2 tx_serial_clk3 tx_serial_clk4	Inputs	Clocks	Serial Clock inputs from the TX PLL. The frequency of this clock depends on the data rate and clock division factor.
tx_pma_clkout	Output	Clock	This is the low speed parallel clock from the TX PMA. It is available when you turn on **Enable tx_pma_clkout port** in the Transceiver Native PHY IP GUI.
tx_pma_div_clkout	Output	Clock	This is a divided version of the recovered parallel clock. It is available when you turn on **Enable tx_pma_div_clkout port** in the Transceiver Native PHY IP GUI.
tx_pma_elecidle[<n>-1:0]	Input	Asynchronous	When asserted this signal forces the transmitter to electrical idle. This port has no effect when you configure the transceiver for the PCI Express protocol.

Name	Direction	Clock Domain	Description
tx_pma_qpipullup[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_qpipullup port (QPI)** in the Transceiver Native PHY IP GUI. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_qpipulldn[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_qpipulldn port (QPI)** in the Transceiver Native PHY IP GUI. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_txdetectrx[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable tx_pma_txdetectrx port (QPI)** in the Transceiver Native PHY IP GUI. When asserted, the receiver detect block in TX PMA detects the presence of a transmitter at the other end of the channel. After receiving tx_pma_txdetectrx request the receiver detect block initiates the detection process. It is only used for Quick Path Interconnect (QPI) applications.
tx_pma_rxfound[<n>-1:0]	Output	Asynchronous	This port is available if you turn on **Enable tx_rxfound_pma port (QPI)** in the GUI. When asserted, indicates that the receiver detect block in TX PMA has detected a transmitter at the other end of the channel. It is only used for Quick Path Interconnect (QPI) applications.

Table 2-42: RX Standard PMA: Data and Optional PMA Ports

Name	Direction	Clock Domain	Description
rx_serial_data[<n>-1:0]	Input	N/A	Serial data input to the RX PMA.
rx_cdr_refclk0	Input	Clock	Reference clock input to the RX clock data recovery (CDR) circuitry.
Optional Ports			
rx_cdr_refclk1– rx_cdr_refclk4	Input	Clock	Reference clock inputs to the RX clock data recovery (CDR) circuitry.
rx_pma_qpipullup[<n>-1:0]	Input	Asynchronous	This port is only used for Quick Path Interconnect (QPI) applications.
rx_is_lockedto-data[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the CDR PLL is locked to the incoming data, rx_serial_data.
rx_is_lockedtoref[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the CDR PLL is locked to the input reference clock.

Name	Direction	Clock Domain	Description
rx_set_lockedto-data[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_set_lockedtoref[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_seriallpbken[<n>-1:0]	Input	Asynchronous	When asserted, enables the TX to RX serial loopback path in the transceiver.
rx_prbs_done[<n>-1:0]	Output	rx_coreclkin	When asserted, indicates the verifier has aligned and captured consecutive PRBS patterns and the first pass through a polynomial is complete. The generator has restarted the polynomial.
rx_prbs_err[<n>-1:0]	Output	rx_coreclkin	When asserted, indicates an error only after the rx_prbs_done signal has been asserted. This signal pulses for every error that occurs. Errors can only occur once per word.
rx_prbs_err_clr[<n>-1:0]	Input	rx_coreclkin	When asserted, clears the PRBS pattern and deasserts the rx_prbs_done signal.
rx_pma_clkout	Output	Clock	This is the recovered parallel from the RX CDR circuitry.
rx_pma_div_clkout	Output	Clock	This is a divided version the recovered RX parallel clock. For the parallel clock, you can divide rx_pma_div_clkout by 1 or 2. For the serial clock, you can divide by 33, 40, 50, or 66. This clock drives rx_coreclkin for different gear box ratios.
rx_pma_clkslip	Input	Clock	A rising edge on this signal causes the RX deserializer to slip the serial data by one clock cycle (2 UI).

Table 2-43: RX Standard PCS: Data, Control, Status, and Clocks

Name	Direction	Clock Domain	Description
rx_parallel_data[<n>128-1:0]	Output	rx_clkout or tx_coreclkin	RX parallel data from the RX PCS to the FPGA fabric. For each 128-bit word of rx_parallel_data, the data bits correspond to rx_parallel_data[7:0] when 8B/10B decoder is enabled and rx_parallel_data[9:0] when 8B/10B decoder is disabled.
unused_rx_parallel_data	Output	rx_clkout or tx_coreclkin	This signal specifies the unused data when you turn on **Enable simplified data interface**.

Name	Direction	Clock Domain	Description
rx_clkout	Output	Clock	FPGA fabric transceiver clock. This clock times rx_parallel_data from the RX PCS to the FPGA fabric.
rx_coreclkin	Input	Clock	RX parallel clock that drives the read side clock of the RX FIFO.

Table 2-44: TX and RX FIFO

Name	Direction	Clock Domain	Description
tx_std_pcfifo_full[<n>-1:0]	Output	tx_clkout or tx_coreclkin.	Indicates when the standard TX FIFO reaches the full threshold.
tx_std_pcfifo_empty[<n>-1:0]	Output	tx_clkout or tx_coreclkin.	Indicates when the standard TX FIFO reaches the empty threshold.
rx_std_pcfifo_full[<n>-1:0]	Output	rx_clkout or tx_coreclkin.	Indicates when the standard RX FIFO reaches the full threshold.
rx_std_pcfifo_empty[<n>-1:0]	Output	rx_clkout or tx_coreclkin.	Indicates when the standard TX FIFO reaches the empty threshold.

Table 2-45: Rate Match FIFO

Name	Direction	Clock Domain	Description
rx_std_rmfifo_full[<n>-1:0]	Output	Asynchronous	Rate match FIFO full flag. When asserted the rate match FIFO is full. You must synchronize this signal. This port is only used for GigE mode.
rx_std_rmfifo_empty[<n>-1:0]	Output	Asynchronous	Rate match FIFO empty flag. When asserted, match FIFO is full. You must synchronize this signal. This port is only used for GigE mode.

Name	Direction	Clock Domain	Description
rx_rmfifostatus[<n>-1:0]	Output	Asynchronous	Indicates FIFO status. The following encodings are defined: • 2'b00: Normal operation • 2'b01: Deletion, rx_std_rmfifo_full = 1 • 2'b10: Insertion, rx_std_rmfifo_empty = 1 • 2'b11: Full. rx_rmfifostatus is a part of rx_parallel_data. rx_rmfifostatus corresponds to rx_parallel_data[14:13].

Table 2-46: 8B/10B Encoder and Decoder

Name	Direction	Clock Domain	Description
tx_forcedisp[<n>(<w>/<s>-1:0]	Input	Asynchronous	This signal allows you to force the disparity of the 8B/10B encoder. When 1'b1, forces the disparity of the output data to the value driven on tx_dispval. When 1'b0, the current running disparity continues. tx_forcedisp is a part of tx_parallel_data. tx_forcedisp corresponds to tx_parallel_data[9].
tx_dispval[<n>(<w>/<s>-1:0]	Input	Asynchronous	Specifies the disparity of the data. tx_dispval is a part of tx_parallel_data. tx_dispval corresponds to tx_dispval[10].
rx_datak[<n><w>/<s>-1:0]	Input	rx_clkout	When 1, indicates that the 8B/10B decoded word of rx_parallel_data is data. When 0, indicates that the 8B/10B decoded word of rx_parallel_data is control. rx_datak is part of rx_parallel_data. For each 128-bit word, rx_datak corresponds to rx_parllel_data[8].

Name	Direction	Clock Domain	Description
rx_errdetect[<n><w>/<s>-1:0]	Output	rx_clkout or rx_coreclkin	When asserted, indicates a code group violation detected on the received code group. Used along with rx_disperr signal to differentiate between code group violation and disparity errors. the following encodings are defined for rx_errdetect/rx_disperr: • 2'b00: no error • 2'b10: code group violation • 2'b11: disparity error. rx_errdetect is a part of rx_parallel_data. For each 128-bit word, rx_errdetect corresponds to rx_parallel_data[9].
rx_disperr[<n><w>/<s>-1:0]	Output	rx_clkout or rx_coreclkin	When asserted, indicates a disparity error on the received code group. rx_disperr is a part of rx_parallel_data. For each 128-bit word, rx_disperr corresponds to rx_parallel_data[11].
rx_runningdisp[<n><w>/<s>-1:0]	Output	rx_clkout or rx_coreclkin	When high, indicates that rx_parallel_data was received with negative disparity. When low, indicates that rx_parallel_data was received with positive disparity. rx_runningdisp is a part of rx_parallel_data. For each 128 bit word, rx_runningdisp corresponds to rx_parallel_data[15].
rx_patterndetect[<n><w>/<s>-1:0]	Output	Asynchronous	When asserted, indicates that the programmed word alignment pattern has been detected in the current word boundary. rx_patterndetect is a part of rx_parallel_data. For each 128-bit word, rx_patterndetect corresponds to rx_parallel_data[12].
rx_syncstatus[<n><w>/<s>-1:0]	Output	Asynchronous	When asserted, indicates that the conditions required for synchronization are being met. rx_syncstatus is a part of rx_parallel_data. For each 128-bit word, rx_syncstatus corresponds to rx_parallel_data[10].

Table 2-47: Word Aligner and Bitslip

Name	Direction	Clock Domain	Description

Name	Direction	Clock Domain	Description
tx_std_bitslipbound-arysel[5 <n>-1:0]	Input	Asynchronous	Bitslip boundary selection signal. Specifies the number of bits that the TX bit slipper must slip.
rx_std_bitslipbound-arysel[5 <n>-1:0]	Output	Asynchronous	This signal operates when the word aligner is in bitslip word alignment mode. It reports the number of bits that the RX block slipped to achieve deterministic latency.
rx_std_wa_ patternalign[<n>-1:0]	Input	Asynchronous	Active when you place the word aligner in manual mode. In manual mode, you align words by asserting rx_std_wa_ patternalign. When the PCS-PMA Interface width is 10 bits, rx_std_wa_ patternalign is level sensitive. For all the other PCS-PMA Interface widths, rx_std_ wa_patternalign is edge sensitive.
rx_std_wa_ala2size[<n>-1:0]	Input	Asynchronous	Used for the SONET protocol. Assert when the A1 and A2 framing bytes must be detected. A1 and A2 are SONET backplane bytes and are only used when the PMA data width is 8 bits.
rx_bitslip[<n>-1:0]	Input	Asynchronous	Used when word aligner mode is bitslip mode. When the Word Aligner is in either Manual (PLD controlled), Synchronous State Machine or Deterministic Latency ,the rx_bitslip signal is not valid and should be tied to 0. For every rising edge of the rx_std_ bitslip signal, the word boundary is shifted by 1 bit. Each bitslip removes the earliest received bit from the received data.

Table 2-48: Bit and Byte Reversal and Polarity Inversion

Name	Direction	Clock Domain	Description
rx_std_byterev_ena[<n>-1:0]	Input	Asynchronous	This control signal is available when the PMA width is 16 or 20 bits. When asserted, enables byte reversal on the RX interface. Used if the MSB and LSB of the transmitted data are erroneously swapped.

Name	Direction	Clock Domain	Description
rx_std_bitrev_ena[<n>-1:0]	Input	Asynchronous	When asserted, enables bit reversal on the RX interface. Bit order may be reversed if external transmission circuitry transmits the most significant bit first. When enabled, the receive circuitry receives all words in the reverse order. The bit reversal circuitry operates on the output of the word aligner. Rewires D[7:0] to D[0:7], and so on.
tx_polinv[<n>-1:0]	Input	Asynchronous	When asserted, the TX polarity bit is inverted. Only active when TX bit polarity inversion is enabled.
rx_polinv[<n>-1:0]	Input	Asynchronous	When asserted, the RX polarity bit is inverted. Only active when RX bit polarity inversion is enabled.

Table 2-49: Signal Detection

Name	Direction	Clock Domain	Description
rx_std_signaldetect[<n>-1:0]	Output	Asynchronous	When enabled, the signal threshold detection circuitry senses whether the signal level present at the RX input buffer is above the signal detect threshold voltage that specified. You can specify the signal detect threshold using a Quartus II Settings File (**.qsf**) assignment. This signal is required for the PCI Express and SATA protocols.
rx_std_elecidle[<n>-1:0]	Input	Asynchronous	When asserted this signal forces the transmitter to electrical idle. This port has no effect when you configure the transceiver for the PCI Express protocol.

Related Information

- **ATX PLL IP** on page 3-6
- **CMU PLL IP** on page 3-21
- **fPLL IP** on page 3-13
- **Ports and Parameters** on page 6-1

- **Transceiver PHY Reset Controller Interfaces** on page 4-12

This section describes the top-level signals for the Transceiver PHY Reset Controller IP core.

Preset Configuration Options

You can select preset settings for the Transceiver Native PHY IP defined for each protocol. Use presets as a starting point to specify parameters for your specific protocol or application.

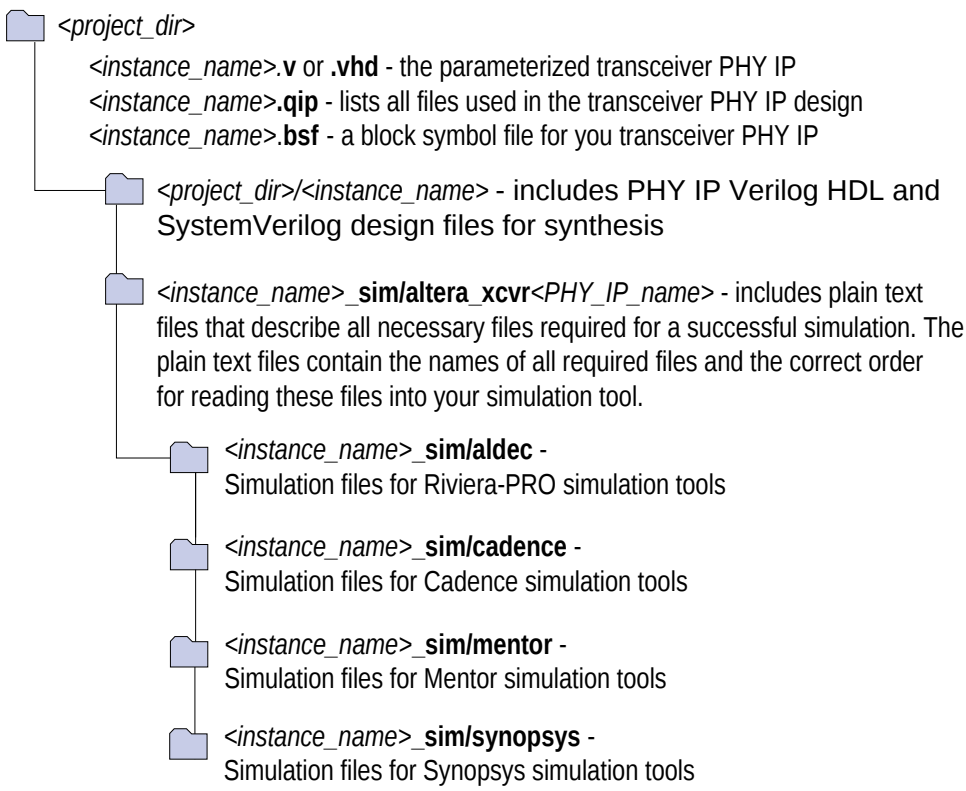
To apply a preset to the Transceiver Native PHY IP, double-click on the preset name. When you apply a preset, all relevant options and parameters are set in the current instance of the Transceiver Native PHY IP. For example, selecting the **Interlaken** preset enables all parameters and ports that the Interlaken protocol requires. The preset does not validate ports or parameters after the preset has been applied.

Selecting a preset does not prevent you from changing any parameter to meet the requirements of your design. Any changes that you make are validated by the design rules for the **Transceiver configuration rules** you specified, not the selected preset.

IP Core File Locations

When you generate your Transceiver Native PHY IP, the Quartus® II software generates the HDL files that define your instance of the IP. In addition, the Quartus II software generates an example Tcl script to compile and simulate your design in the ModelSim simulator.

Figure 2-8: Directory Structure for Generated Files



The following table describes the directories and the most important files for the parameterized Transceiver Native PHY IP core and the simulation environment. These files are in clear text.

Table 2-50: Transceiver Native PHY Files and Directories

File Name	Description
<project_dir>	The top-level project directory.
<instance_name> **.v** or **.vhd**	The top-level design file.
<instance_name> **.qip**	A list of all files necessary for Quartus II compilation.
<instance_name> **.bsf**	A Block Symbol File (.bsf) for your Transceiver Native PHY instance.
<project_dir>/<instance_name>/	The directory that stores the HDL files that define the protocol-specific Transceiver Native PHY IP. These files are used for synthesis.
<project_dir>/<instance_name> **_sim/ altera_xcvr_** *<PHY_IP_name>/*	The simulation directory.
<project_dir>/<instance_name>_sim/ **aldec**	Simulation files for Riviera-PRO simulation tools.
<project_dir>/<instance_name>_sim/ **cadence**	Simulation files for Cadence simulation tools.
<project_dir>/<instance_name>_sim/ **mentor**	Simulation files for Mentor simulation tools.
<project_dir>/<instance_name>_sim/ **synopsys**	Simulation files for Synopsys simulation tools.

The Verilog and VHDL Transceiver Native PHY IP cores have been tested with the following simulators:

- ModelSim SE
- Synopsys VCS MX
- Cadence NCSim

If you select VHDL for your transceiver PHY, only the wrapper generated by the Quartus II software is in VHDL. All the underlying files are written in Verilog or System Verilog. To enable simulation using a VHDL-only ModelSim license, the underlying Verilog and System Verilog files for the Transceiver Native PHY IP are encrypted so that they can be used with the top-level VHDL wrapper without using a mixed-language simulator.

For more information about simulating with ModelSim, refer to the *Mentor Graphics ModelSim Support* chapter in volume 3 of the *Quartus II Handbook*.

The Transceiver Native PHY IP cores do not support the NativeLink feature in the Quartus II software.

Related Information

- **Mentor Graphics ModelSim Support**
- **Simulating the Transceiver Native PHY IP Core** on page 2-256

Interlaken

Interlaken is a scalable, channelized chip-to-chip interconnect protocol.

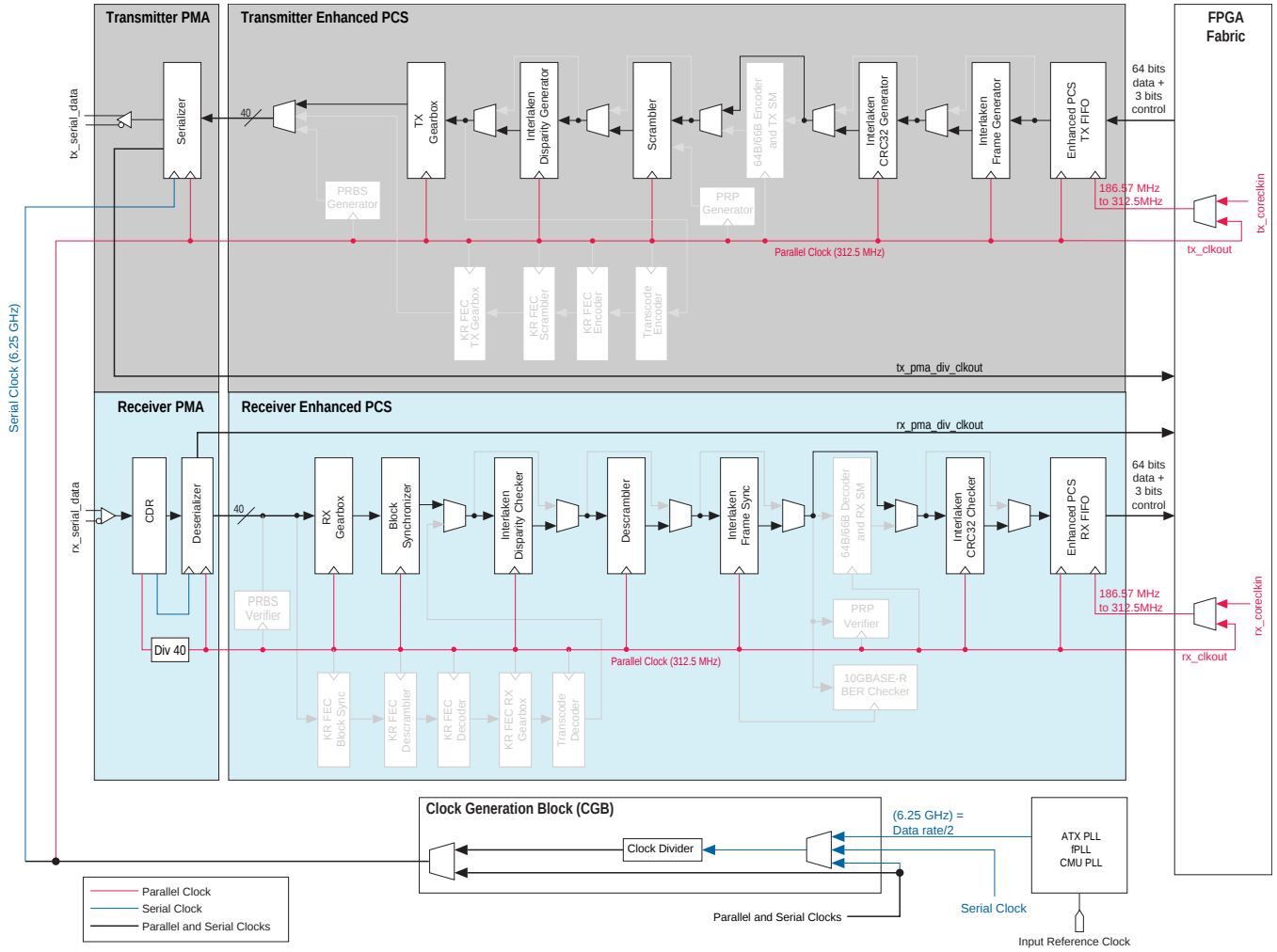
The key advantages of Interlaken are scalability and its low I/O count compared to earlier protocols such as SPI 4.2. Other key features include flow control, low overhead framing, and extensive integrity checking.

Interlaken operates on 64-bit data words and 3 control bits, which are striped round-robin across the lanes. The protocol accepts packets on 256 logical channels and is expandable to accommodate up to 65,536 logical channels. Packets are split into small bursts that can optionally be interleaved. The burst semantics include integrity checking and per logical channel flow control.

The Interlaken interface is supported with 1 to 48 lanes running at data rates up to 17.4 Gbps per lane. Interlaken is implemented using the Enhanced PCS. The Enhanced PCS has demonstrated interoperability with Interlaken ASSP vendors and third-party IP suppliers.

Figure 2-9: Transceiver Channel Datapath and Clocking for Interlaken

This assumes the serial data rate is 12.5 Gbps and the PMA width is 40 bits.



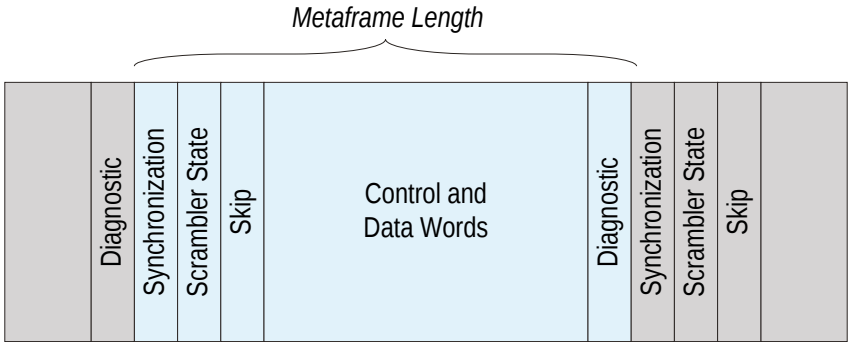
Related Information

- **Interlaken Protocol Definition v1.2**
- **Interlaken Look-Aside Protocol Definition, v1.1**

Metaframe Format and Framing Layer Control Word

The Enhanced PCS supports programmable metaframe lengths from 5 to 8192 words. However, for stability and performance, Altera recommends you set the frame length to no less than 128 words. In simulation, use a smaller metaframe length to reduce simulation times. The payload of a metaframe could be pure payload and Burst/Idle control word from MAC layer.

Figure 2-10: Framing Layer Metaframe Format



The framing control words include:

- Synchronization (SYNC)—for frame delineation and lane alignment (deskew)
- Scrambler State (SCRM)—to synchronize the scrambler
- Skip (SKIP)—for clock compensation in a repeater
- Diagnostic (DIAG)—provides per-lane error check and optional status message

To form a metaframe, the Enhanced PCS frame generator inserts the framing control words and encapsulates the control and data words read from the TX FIFO as the metaframe payload.

Figure 2-11: Interlaken Synchronization and Scrambler State Words Format

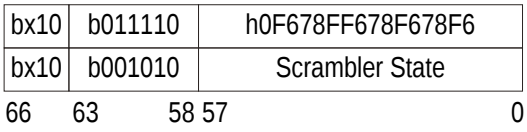
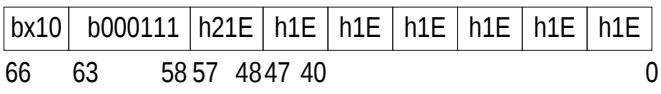


Figure 2-12: Interlaken Skip Word Format

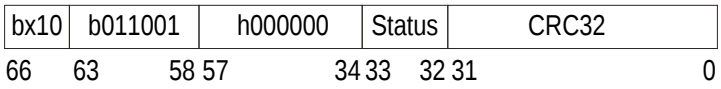


The DIAG word is comprised of a status field and a CRC-32 field. The 2-bit status is defined by the Interlaken specification as:

- Bit 1 (Bit 33): Lane health
  - 1: lane is healthy
  - 0: lane is not healthy
- Bit 0 (Bit 32): Link health
  - 1: Link is healthy
  - 0: Link is not healthy

The `tx_enh_frame_diag_status[1:0]` input from the FPGA fabric is inserted into the Status field each time a DIAG word is created by the framing generator.

Figure 2-13: Interlaken Diagnostic Word



Interlaken Configuration Clocking and Bonding

The Arria 10 Interlaken PHY layer solution is scalable and has flexible data rates. You can implement a single lane link or bond up to 48 lanes together. You can choose a lane data rate up to 17.4 Gbps. You can also choose between different reference clock frequencies, depending on the PLL used to clock the transceiver. Refer to the *Arria 10 Device Datasheet* for the minimum and maximum data rates that Arria 10 transceivers can support at different speed grades.

You can use an ATX PLL or fPLL to provide the clock for the transmit channel. An ATX PLL has better jitter performance compared to an fPLL. You can use the CMU PLL to clock only the non-bonded Interlaken implementation. However, if you use the CMU PLL, you lose one RX transceiver channel.

For the multi-lane Interlaken interface, TX channels are usually bonded together to minimize the transmit skew between all bonded channels. Currently, xN bonding and a PLL feedback compensation bonding scheme are available to support a multi-lane Interlaken implementation. If the system tolerates higher channel-to-channel skew, you can choose to not bond the TX channels.

To implement bonded multi-channel Interlaken, all channels must be placed contiguously. The channels may all be placed in one bank (if not greater than six lanes) or they may span several banks.

Related Information

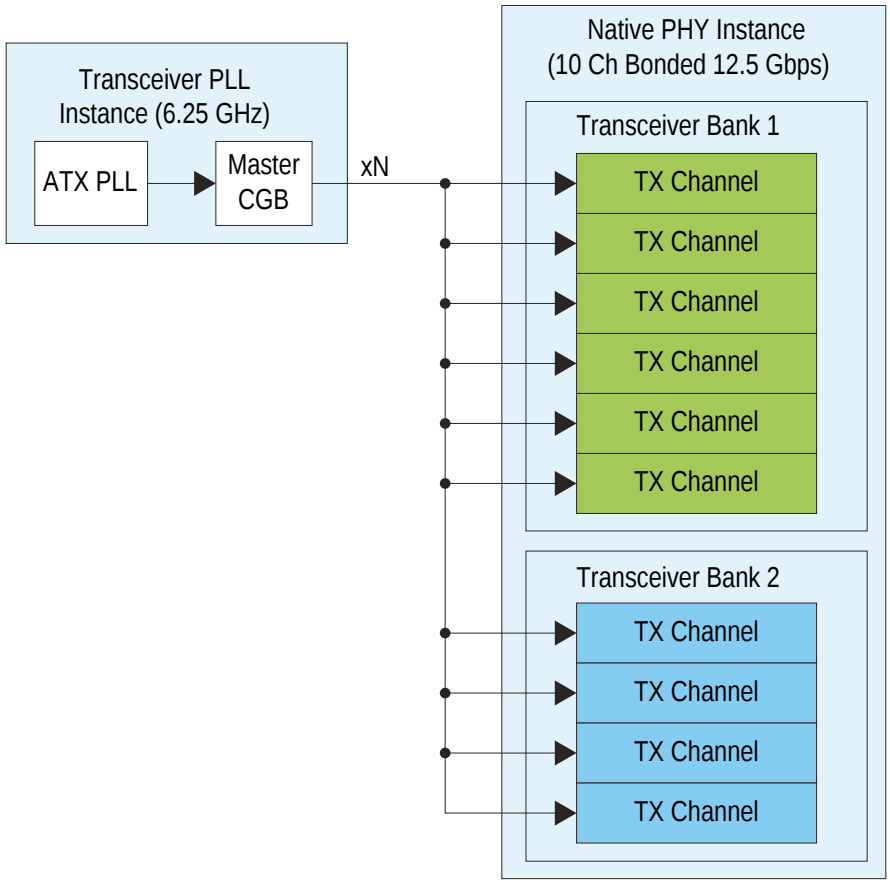
- **Arria 10 Device Datasheet**
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks

xN Clock Bonding Scenario

The following figure shows a xN bonding example supporting 10 lanes. Each lane is running at 12.5 Gbps. The first six TX channels reside in one transceiver bank and the other four TX channels reside in the adjacent transceiver bank. The ATX PLL provides the serial clock to the master CGB. The CGB then provides parallel and serial clocks to all of the TX channels inside the same bank and other banks through the xN clock network.

Because of xN clock network skew, the maximum achievable data rate decreases when TX channels span several transceiver banks.

Figure 2-14: 10X12.5 Gbps xN Bonding



Related Information

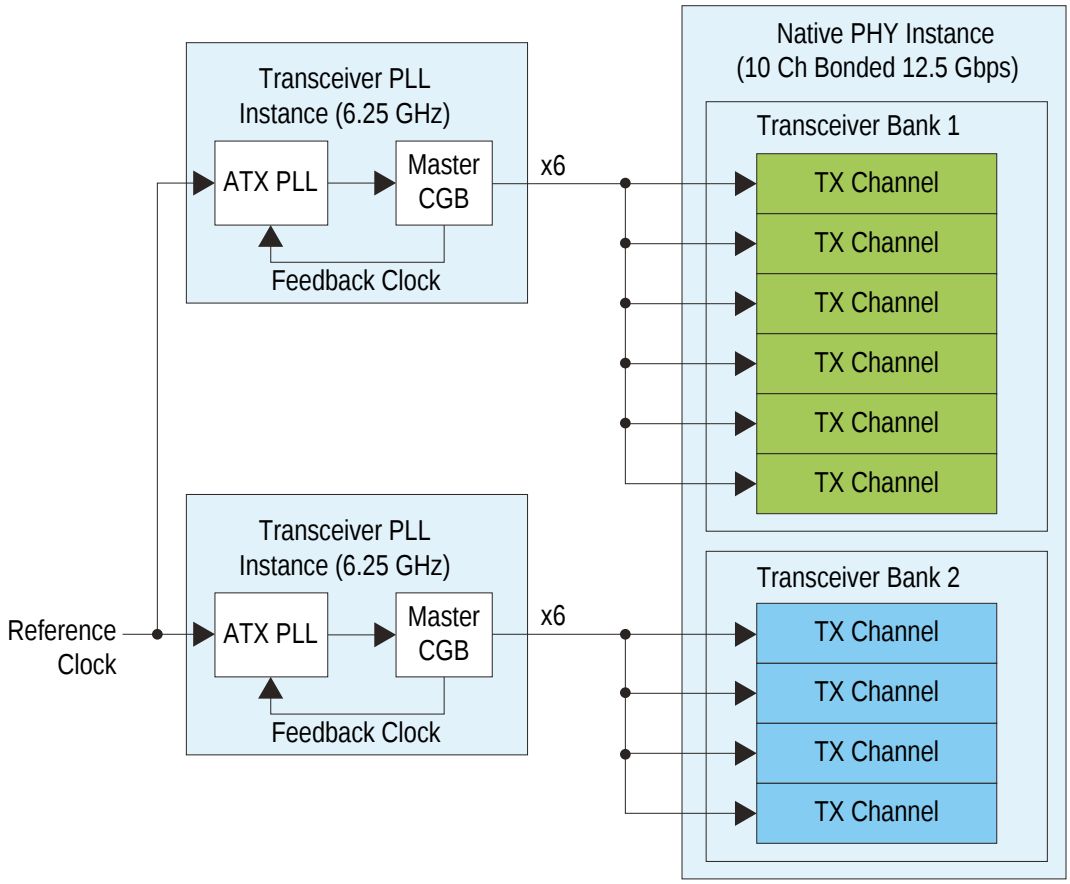
- **x6/xN Bonding Mode** on page 3-45
  For detailed information on xN bonding limitations
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks

PLL Feedback Compensation Clock Bonding Scenario

In the following figure, each lane is running at 12.5 Gbps. The first six TX channels reside in one transceiver bank and the other four TX channels reside in the adjacent transceiver bank. The difference between feedback compensation bonding and xN bonding is that feedback compensation bonding separates the TX channels into multiple bonding groups, each group being driven by a separate x6 clock network. In feedback compensation bonding, the separate x6 clocks are in phase and frequency aligned with each other. One PLL from each transceiver bank drives the clock to master CGB and then drives these clocks to TX channels that reside in the same bank only. In xN bonding, all channels are driven by the xN clock network. The data rate decrease imposed by xN bonding does not apply to PLL feedback compensation bonding.

The feedback to the PLL for each bonded group is the parallel clock from the master CGB, which has the same frequency as `tx_clkout`. The reference clock for the PLL must match the frequency of this feedback clock. For example, given that the Interlaken interface runs at 12.5Gbps per lane, and PCS-PMA width is 40 bits, the only available frequency of the reference clock is 312.5 MHz.

Figure 2-15: 10X12.5 Gbps PLL Feedback Compensation Bonding



Related Information

PLL Feedback Compensation Bonding Mode on page 3-47
For other limitations on feedback compensation bonding

TX Multi-Lane Bonding and RX Multi-Lane Deskew Alignment State Machine

The Interlaken configuration sets the enhanced PCS TX and RX FIFOs in Interlaken elastic buffer mode. In this mode of operation, TX and RX FIFO control and status port signals are provided to the FPGA fabric. Connect these signals to the MAC layer as required by the protocol. Based on these FIFO status and control signals, you can implement the multi-lane deskew alignment state machine in the FPGA fabric to control the transceiver RX FIFO block. You must also implement the soft bonding logic to control the transceiver TX FIFO block.

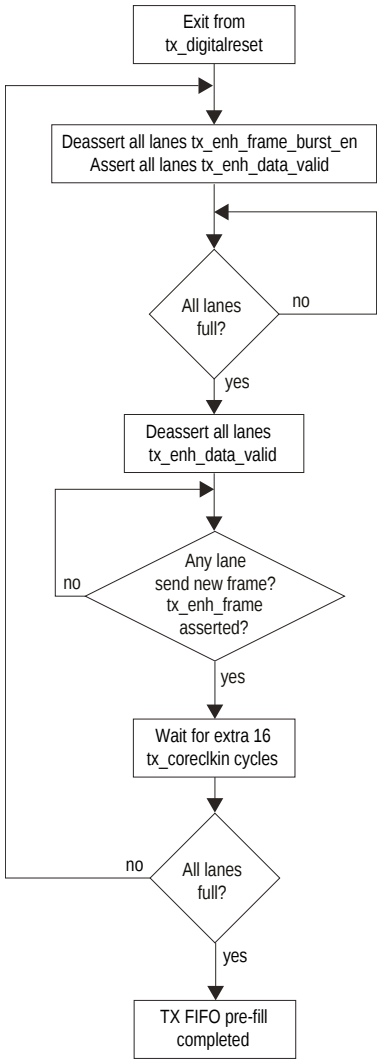
TX FIFO Soft Bonding

The MAC layer logic and TX soft bonding logic control the writing of the Interlaken word to the TX FIFO with `tx_enh_data_valid` (functions as a TX FIFO write enable) by monitoring the TX FIFO flags(`tx_enh_fifo_full`, `tx_enh_fifo_pfull`, `tx_enh_fifo_empty`, `tx_enh_fifo_pempty`,

tx_enh_fifo_cnt, and so forth). On the TX FIFO read side, a read enable is controlled by the frame generator. If tx_enh_frame_burst_en is asserted high, the frame generator reads data from the TX FIFO.

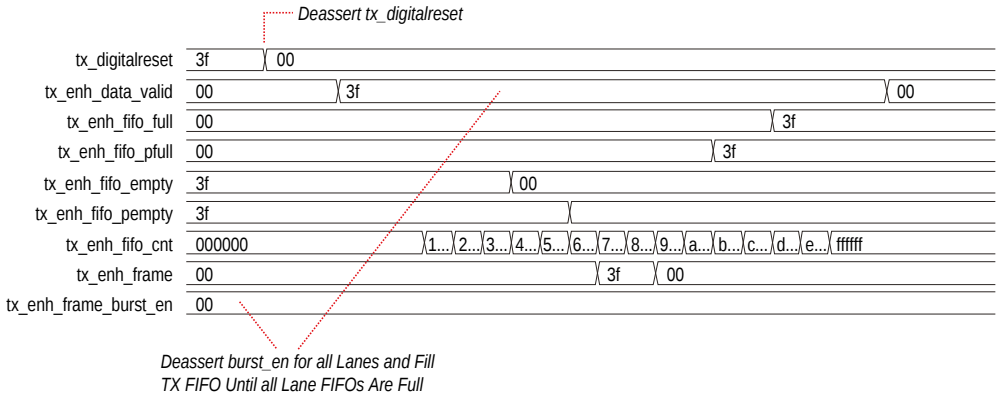
A TX FIFO pre-fill stage must be implemented to perform the TX channel soft bonding. The following figure shows the state of the pre-fill process.

Figure 2-16: TX Soft Bonding Flow



The following figure shows that after the deassertion of tx_digitalreset, TX soft bonding logic starts filling the TX FIFO until all lanes are full.

Figure 2-17: TX FIFO Pre-fill

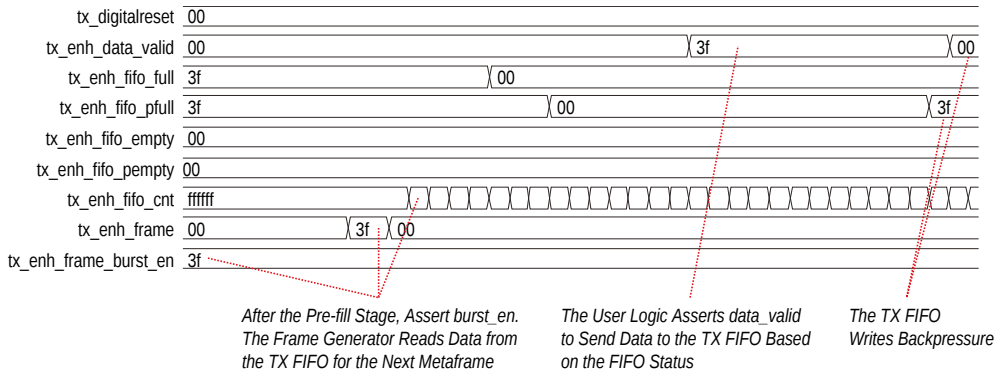


After the TX FIFO pre-fill stage completes, the transmit lanes are synchronized and the MAC layer can begin to send valid data to the transceiver's TX FIFO. You must never allow the TX FIFO to overflow or underflow. If it does, you must reset the transceiver and repeat the TX FIFO pre-fill stage.

For a single lane Interlaken implementation, TX FIFO soft bonding is not required. You can begin sending an Interlaken word to the TX FIFO after the deassertion of `tx_digitalreset`.

The following figure shows the MAC layer sending valid data to the Native PHY after the pre-fill stage. `tx_enh_frame_burst_en` is asserted, allowing the frame generator to read data from the TX FIFO. The TX MAC layer can now control `tx_enh_data_valid` to write data to the TX FIFO based on the FIFO status signals.

Figure 2-18: MAC Sending Valid Data


tx_digitalreset
tx_enh_data_valid
tx_enh_fifo_full
tx_enh_fifo_pfull
tx_enh_fifo_empty
tx_enh_fifo_pempty
tx_enh_fifo_cnt
tx_enh_frame
tx_enh_frame_burst_en
After the Pre-fill Stage, Assert burst_en. The Frame Generator Reads Data from the TX FIFO for the Next Metaframe
The User Logic Asserts data_valid to Send Data to the TX FIFO Based on the FIFO Status
The TX FIFO Writes Backpressure

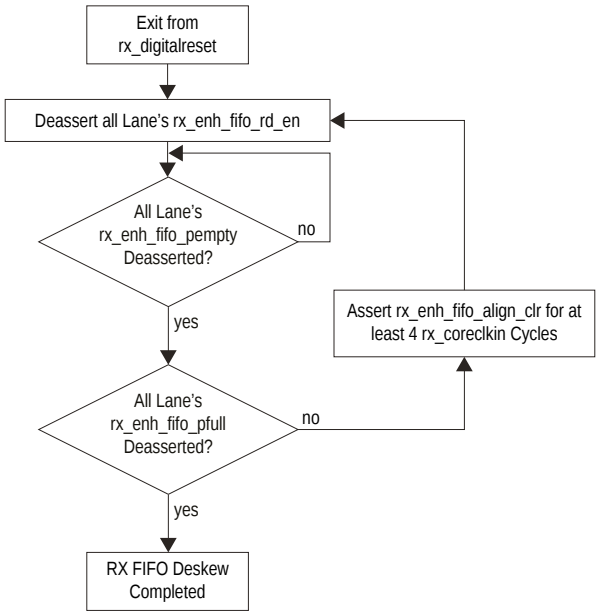
RX Multi-lane FIFO Deskew State Machine

Deskew logic is required at the receiver side to eliminate the lane-to-lane skew created at the transmitter of the link partner, PCB, medium, and local receiver PMA.

You can implement a multi-lane alignment deskew state machine to control the RX FIFO operation based on available RX FIFO status flags and control signals.

The following figure shows the state flow of RX FIFO deskew.

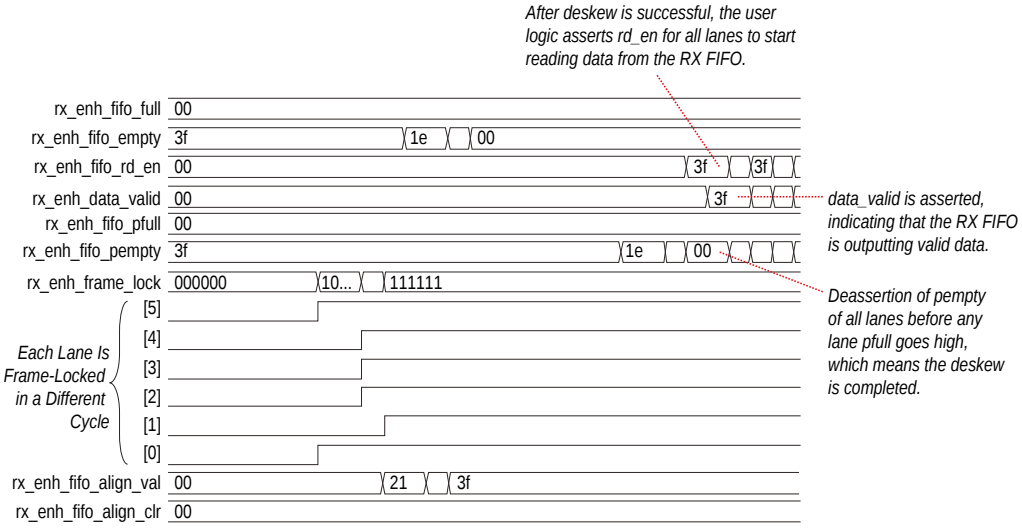
Figure 2-19: RX FIFO Deskew Flow



Each lane's `rx_enh_fifo_rd_en` should remain deasserted before the RX FIFO deskew is completed. After frame lock is achieved (indicated by the assertion of `rx_enh_frame_lock`; this signal is not shown in the above state flow), data is written into the RX FIFO after the first alignment word (SYNC word) is found on that channel. Accordingly, the RX FIFO partially empty flag (`rx_enh_fifo_pempty`) of that channel is asserted. The state machine monitors the `rx_enh_fifo_pempty` and `rx_enh_fifo_pfull` signals of all channels. If the `rx_enh_fifo_pempty` signals from all channels deassert before any channels `rx_enh_fifo_pfull` assert, which implies the SYNC word has been found on all lanes of the link, the MAC layer can start reading from all the RX FIFO by asserting `rx_enh_fifo_rd_en` simultaneously. Otherwise, if the `rx_enh_fifo_pfull` signal of any channel asserts high before the `rx_enh_fifo_pempty` signals deassertion on all channels, the state macihne needs to flush the RX FIFO by asserting `rx_enh_fifo_align_clr` high for 4 cycles and repeating the soft deskew process.

The following figure shows one RX deskew scenario. In this scenario, all of the RX FIFO partially empty lanes are deasserted while the pfull lanes are still deasserted. This indicates the deskew is successful and the FPGA fabric starts reading data from the RX FIFO.

Figure 2-20: RX FIFO Deskew



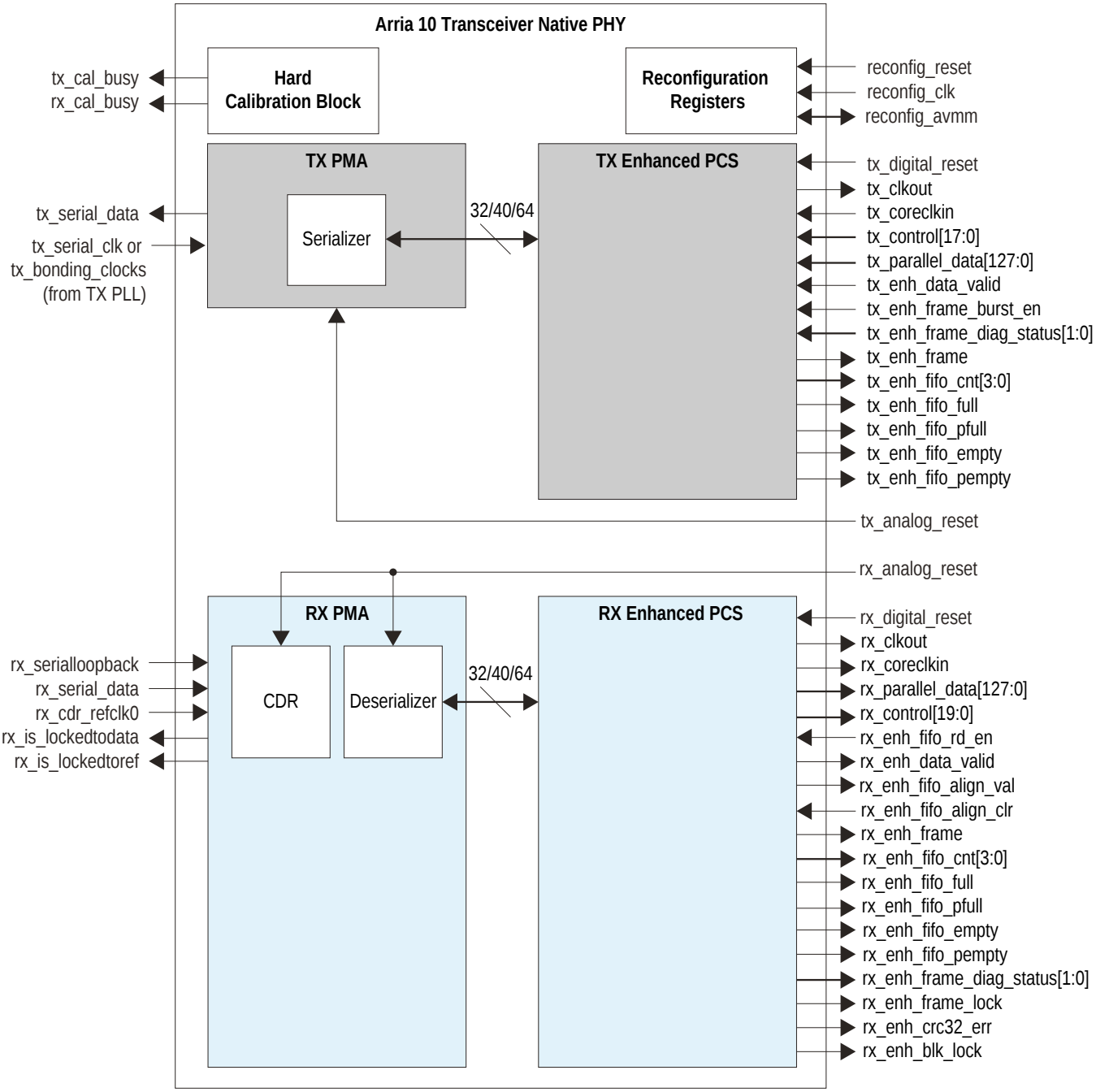
How to Implement Interlaken in Arria 10 Transceivers

Before you begin

You should be familiar with the Interlaken protocol, Enhanced PCS and PMA architecture, PLL architecture, and the reset controller before implementing the Interlaken protocol PHY layer.

1. Open the MegaWizard Plug-In Manager and select the **Arria 10 Transceiver Native PHY IP**. Refer to **Select and Instantiate PHY IP** on page 2-2 for more details.
2. Select **Interlaken** from the **Transceiver configuration rules** list located under **Datapath Options**.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for Interlaken** as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 2-21: Signals and Ports of Native PHY IP for Interlaken

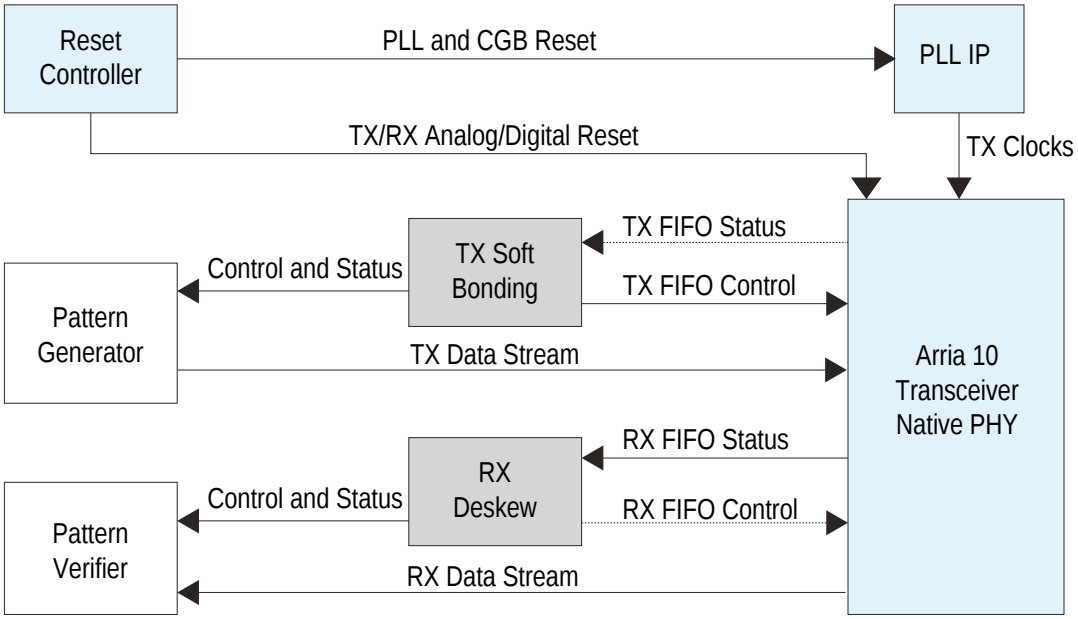


5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Altera Transceiver PHY Reset Controller IP.
7. Implement a TX soft bonding logic and an RX multi-lane alignment deskew state machine using fabric logic resources for multi-lane Interlaken implementation.
8. Connect the Native PHY IP to the PLL IP and the reset controller.

Figure 2-22: Connection Guidelines for an Interlaken PHY Design

This figure shows the connection of all these blocks in the Interlaken PHY design example available on the Altera Wiki website.

For blue blocks, Altera provides a megafunction IP. The gray blocks are the TX soft bonding logic and RX deskew logic that are included in the design example. The white blocks are your test logic or MAC layer logic.



9. Simulate your design to verify its functionality.

Figure 2-23: 24 Lanes Bonded Interlaken Link, TX Direction

To show more details, three different time segments are shown with the same zoom level.

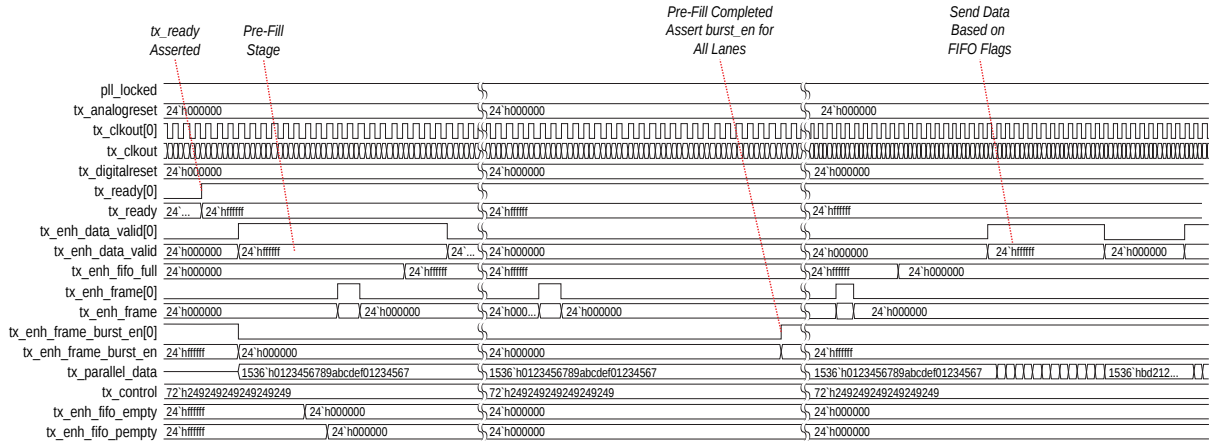
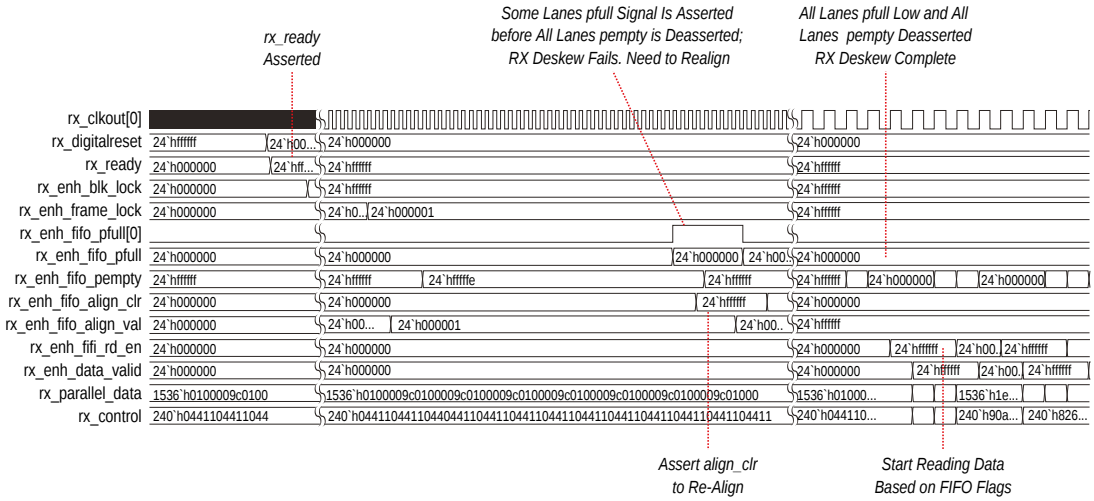


Figure 2-24: 24 Lanes Bonded Interlaken Link, RX Direction

To show more details, three different time segments are shown with different zoom level.



Related Information

- **Arria 10 Enhanced PCS Architecture** on page 5-14
  For more information about Enhanced PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Enhanced PCS and PMA Ports** on page 2-37
  For detailed information about the available ports in the Interlaken protocol.

Design Example

Altera provides a PHY layer-only design example to help you integrate an Interlaken PHY into your complete design.

The TX soft bonding logic and RX multi-lane deskew state machine are included in the design example. Altera recommends that you integrate these two modules into your design.

The *Interlaken Design Example* is available on the Arria 10 Transceiver PHY Design Examples Wiki page.

Note: The design examples on the Wiki page provide useful guidance for developing your own designs, but they are not guaranteed by Altera. Use them with caution.

Related Information

Interlaken Design Example

Native PHY IP Parameter Settings for Interlaken

Table 2-51: General and Datapath Parameters

Parameter	Value
Device speed grade	fastest
Message level for rule violations	error warning
Transceiver Configuration Rules	Interlaken
Transceiver mode	TX / RX Duplex TX Simplex RX Simplex
Number of data channels	1 to 48
Data rate	Up to 17.4 Gbps
Enable reconfiguration between Standard and Enhanced PCS	On / Off
Enable simplified data interface	On / Off

Table 2-52: TX PMA Parameters

Parameter	Value
TX channel bonding mode	Non bonded PMA bonding
TX local clock division factor	1, 2, 4, 8
Number of TX PLLs	1, 2, 3, 4
Main TX PLL logical index	0

Table 2-53: RX PMA Parameters

Parameter	Value
Number of CDR reference clocks	1 to 5
Selected CDR reference clock	0 to 4
Selected CDR reference clock frequency	Select from drop-down menu

Parameter	Value
PPM detector threshold	**62.5**, **100**, **125**, **200**, **250**, **300**, **500**, **1000**

Table 2-54: Enhanced PCS Parameters

Parameter	Value
Enhanced PCS protocol mode	**Interlaken**
Enhanced PCS to PMA interface width	**32**, **40**, **64**
FPGA fabric to Enhanced PCS interface width	**67**
Enable RX/TX FIFO double-width mode	**Off**
TX FIFO mode	**Interlaken**
TX FIFO partially full threshold	from **10**-**13** (no less than pempty_threshold+8)
TX FIFO partially empty threshold	**2** to **5**
Enable tx_enh_fifo_full port	**On** / **Off**
Enable tx_enh_fifo_pfull port	**On** / **Off**
Enable tx_enh_fifo_empty port	**On** / **Off**
Enable tx_enh_fifo_pempty port	**On** / **Off**
Enable tx_enh_fifo_cnt port	**On** / **Off**
RX FIFO mode	**Interlaken**
RX FIFO partially full threshold	from **10**-**29** (no less than pempty_threshold+8)
RX FIFO partially empty threshold	**2** to **10**
Enable RX FIFO alignment word deletion (Interlaken)	**On** / **Off**
Enable RX FIFO control word deletion (Interlaken)	**On** / **Off**
Enable rx_enh_fifo_data_valid port	**On**
Enable rx_enh_fifo_full port	**On** / **Off**
Enable rx_enh_fifo_pfull port	**On** / **Off**
Enable rx_enh_fifo_empty port	**On** / **Off**
Enable rx_enh_fifo_pempty port	**On** / **Off**
Enable rx_enh_fifo_cnt port	**On** / **Off**

Parameter	Value
Enable rx_enh_fifo_rd_en port (Interlaken)	On
Enable rx_enh_fifo_align_val port (Interlaken)	On / Off
Enable rx_enh_fifo_align_clr port (Interlaken)	On

Table 2-55: Interlaken Frame Generator Parameters

Parameter	Value
Enable Interlaken frame generator	On
Frame generator metaframe length	5 to 8192
Enable frame burst	On
Enable tx_enh_frame port	On
Enable tx_enh_frame_diag_status port	On
Enable tx_enh_frame_burst_en port	On

Table 2-56: Interlaken Frame Synchronizer Parameters

Parameter	Value
Enable Interlaken frame synchronizer	On
Frame synchronizer metaframe length	5 to 8192
Enable rx_enh_frame port	On
Enable rx_enh_frame_lock port	On / Off
Enable rx_enh_frame_diag_status port	On / Off

Table 2-57: Interlaken CRC-32 Generator and Checker Parameters

Parameter	Value
Enable Interlaken TX CRC-32 generator	On
Enable TX CRC-32 generator error insertion	On / Off
Enable Interlaken RX CRC-32 checker	On
Enable rx_enh_crc32_err port	On / Off

Table 2-58: 64B / 66B Encoder and Decoder Parameters

Parameter	Value
Enable TX 64B/66B encoder	Off
Enable RX 64B/66B decoder	Off
Enable TX sync header error insertion	Off

Table 2-59: Scrambler and Descrambler Parameters

Parameter	Value
Enable TX scrambler (10GBASE-R / Interlaken)	On
TX scrambler seed (10GBASE-R / Interlaken)	0x1 to 0x3FFFFFFFFFFFFF
Enable RX descrambler (10GBASE-R / Interlaken)	On

Table 2-60: Interlaken Disparity Generator and Checker Parameters

Parameter	Value
Enable Interlaken TX disparity generator	On
Enable Interlaken RX disparity checker	On

Table 2-61: Block Synchronizer Parameters

Parameter	Value
Enable RX block synchronizer	On
Enable rx_enh_blk_lock port	On / Off

Table 2-62: Gearbox Parameters

Parameter	Value
Enable TX data bitslip	On / Off
Enable TX data polarity inversion	On / Off
Enable RX data bitslip	On / Off
Enable RX data polarity inversion	On / Off
Enable tx_enh_bitslip port	On / Off

Parameter	Value
Enable rx_bitslip port	On / Off

Table 2-63: Dynamic Reconfiguration Parameters

Parameter	Value
Enable dynamic reconfiguration	On / Off
Share reconfiguration interface	On / Off
Enable embedded JTAG AVMM master	On / Off

Table 2-64: Configuration Files Parameters

Parameter	Value
Configuration file prefix	—
Generate SystemVerilog package file	On / Off
Generate C header file	On / Off
Generate MIF (Memory Intialization File)	On / Off

Table 2-65: Configuration Profiles Parameters

Parameter	Value
Enable multiple reconfiguration profiles	On / Off
Generate reduced reconfiguration files	On / Off
Number of reconfiguration profiles	2, 3, 4, 5, 6, 7, 8
Selected reconfiguration profile	—

Table 2-66: Generation Options Parameters

Parameter	Value
Generate parameter documentation file	On / Off

Ethernet

Data Rate	Transceiver Configuration Rule/IP
1G	• Gigabit Ethernet • Gigabit Ethernet 1588

Data Rate	Transceiver Configuration Rule/IP
10G	• 10GBASE-R • 10GBASE-R 1588 • 10GBASE-R with KR FEC • 10GBASE-KR PHY IP
1G/10G	1G/10G Ethernet PHY IP

Gigabit Ethernet (GbE) and GbE with 1588

IEEE 802.3 defines Gigabit Ethernet as an intermediate (or transition) layer that interfaces various physical media with the media access control (MAC) in a Gigabit Ethernet system. Gigabit Ethernet PHY shields the MAC layer from the specific nature of the underlying medium and is divided into three sub-layers shown in the following figure.

Figure 2-25: GbE PHY Connection to IEEE802.3 MAC and RS

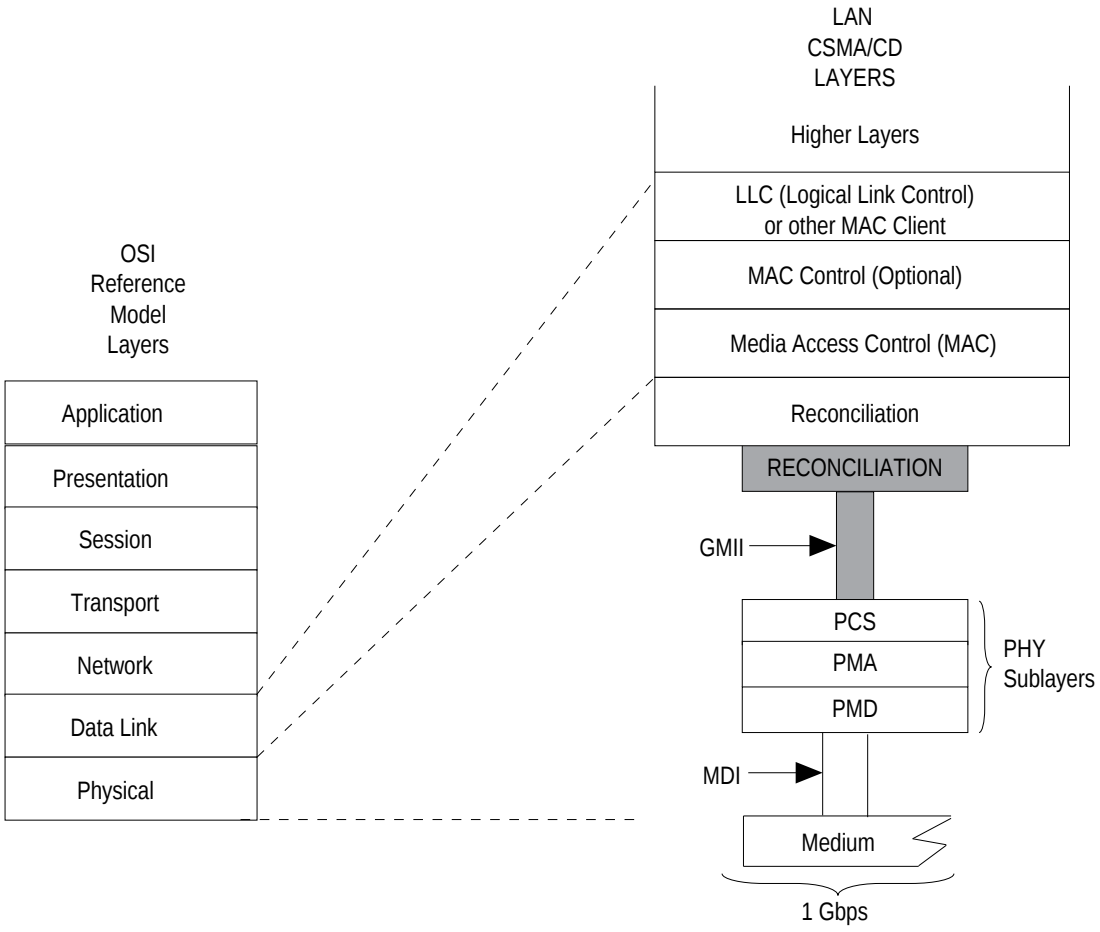
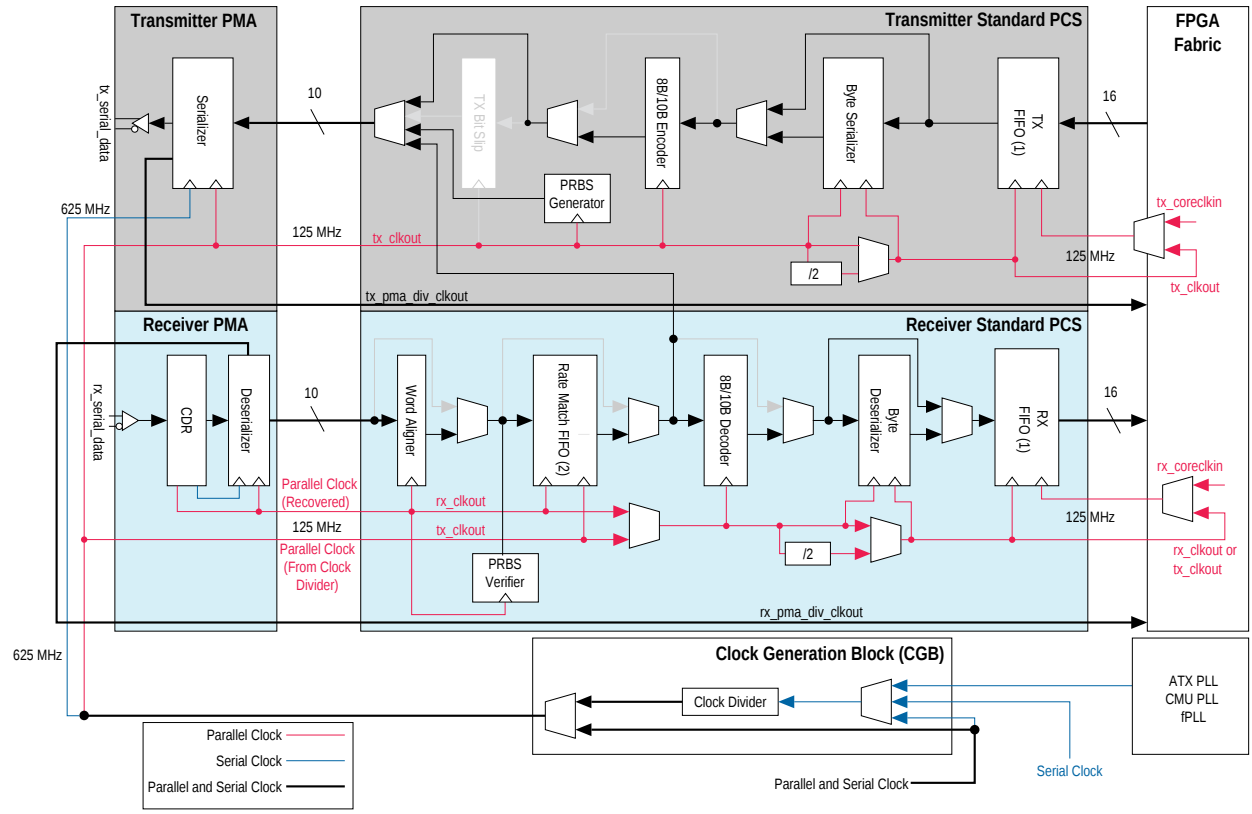


Figure 2-26: Transceiver Channel Datapath and Clocking at 1250 Mbps for GbE, GbE with 1588



Notes:
1. This block is set in **low latency** mode for GbE and **register_fifo** mode for GbE with 1588.
2. Rate match FIFO is disabled for GbE with 1588.

Note: The transceivers do not have built-in support for other PCS functions; for example, the auto-negotiation state machine, collision-detect, and carrier-sense. If required, you must implement these functions in the FPGA fabric or external circuits.

GbE with 1588

GbE with 1588 provides a standard method to synchronize devices on a network with submicrosecond precision. To improve performance, the protocol synchronizes slave clocks to a master clock so that events and time stamps are synchronized in all devices. The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock.

8B/10B Encoding for GbE, GbE with 1588

The 8B/10B encoder clocks 8-bit data and 1-bit control identifiers from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is sent to the PMA.

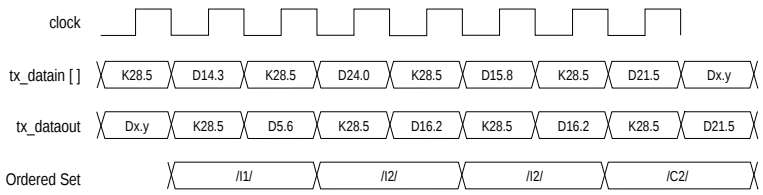
The IEEE 802.3 specification requires GbE to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

For the GbE protocol, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The

exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, an /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Note: /D14.3/, /D24.0/, and /D15.8/ is replaced by /D5.6/ or /D16.2/ (for I1 and I2 ordered sets). D21.5 (/C1/) is not replaced.

Figure 2-27: Idle Ordered-Set Generation Example



Related Information

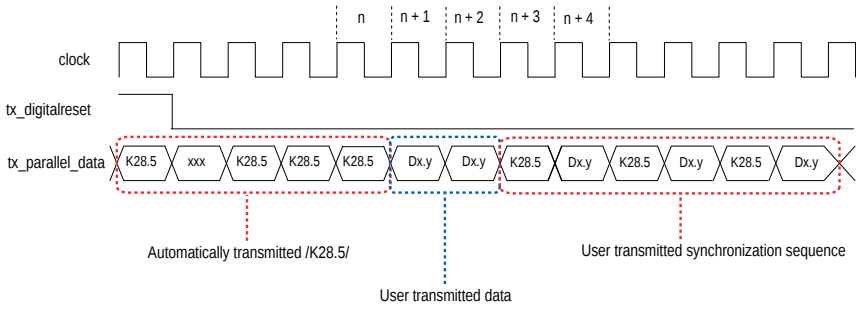
8B/10B Encoder on page 5-33

Reset Condition for 8B/10B Encoder in GbE, GbE with 1588

After deassertion of `tx_digitalreset`, the transmitters automatically transmit at least three /K28.5/ comma code groups before transmitting user data on the `tx_parallel_data` port. This could affect the synchronization state machine behavior at the receiver.

Depending on when you start transmitting the synchronization sequence, there could be an even or odd number of /Dx.y/ code groups transmitted between the last of the three automatically sent /K28.5/ code groups and the first /K28.5/ code group of the synchronization sequence. If there is an even number of /Dx.y/code groups received between these two /K28.5/ code groups, the first /K28.5/ code group of the synchronization sequence begins at an odd code group boundary. The synchronization state machine treats this as an error condition and goes into the loss of sync state.

Figure 2-28: Reset Condition



Word Alignment for GbE, GbE with 1588

The word aligner for the GbE and GbE with 1588 protocols is configured in automatic synchronization state machine mode. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives three consecutive synchronization ordered sets. A synchronization ordered set is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered sets.

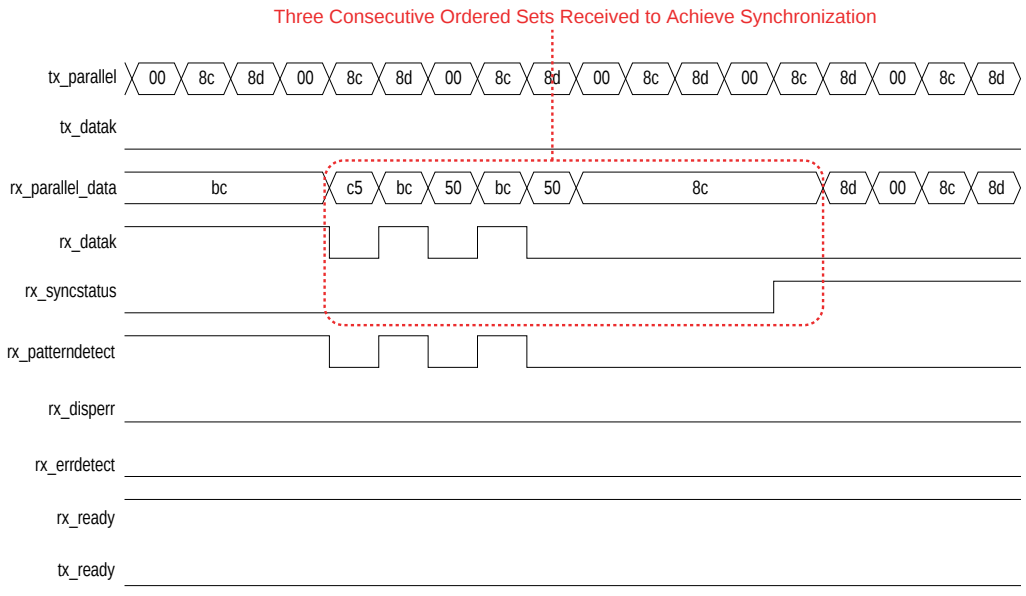
Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that the lane has fallen out of synchronization. The receiver loses synchronization when it detects three invalid code groups separated by less than three valid code groups or when it is reset.

Table 2-67: Synchronization State Machine Parameter Settings for GbE

Synchronization State Machine Parameter	Setting
Number of word alignment patterns to achieve sync	3
Number of invalid data words to lose sync	3
Number of valid data words to decrement error count	3

The following figure shows `rx_syncstatus` high when three consecutive ordered sets are sent through `tx_parallel_data`.

Figure 2-29: rx_syncstatus High



Related Information

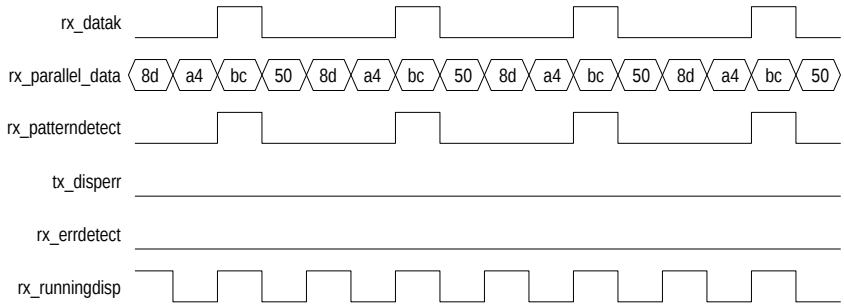
Word Aligner on page 5-37

8B/10B Decoding for GbE, GbE with 1588

The general functionality for the 8B/10B decoder is to take a 10-bit encoded value as input and produce an 8-bit data value and 1-bit control value as output.

The following figure shows Dx.y(8d), Dx.y(a4), K28.5(bc), Dx.y(50) received at `rx_parallel_data`. /K28.5/ is set as the word alignment pattern. `rx_patterndetect` goes high whenever it detects /K28.5/(bc). `rx_datak` is high when bc is received, indicating that the decoded word is a control word. Otherwise, `rx_datak` is low. `rx_runningdisp` is high for 8d, indicating that the decoded word has negative disparity and a4 has positive disparity.

Figure 2-30: Decoding for GbE



Related Information

8B/10B Decoder on page 5-42

Rate Match FIFO for GbE

The rate match FIFO is capable of compensating for up to ±100 ppm (200 ppm total) difference between the upstream transmitter and the local receiver reference clock.

Note: Rate match FIFO is not available in the GbE with 1588 protocol.

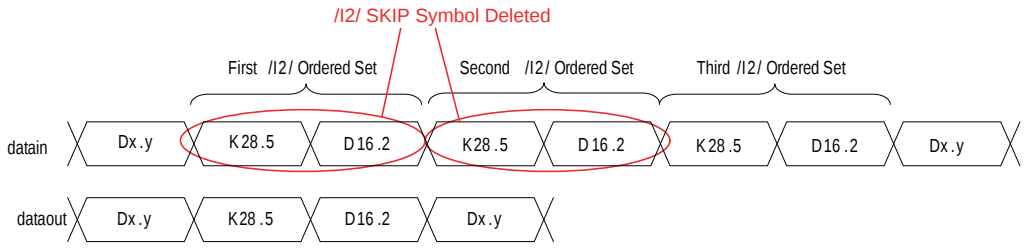
The GbE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate match FIFO deletes or inserts the /I2/ (/K28.5/D16.2/) ordered set to prevent the rate match FIFO from overflowing or under running during normal packet transmission. The rate match FIFO deletes or inserts the first two bytes of the /C2/ ordered set (/K28.5/D2.2/Dx.y/Dx.y/) to prevent the rate match FIFO from overflowing or underflowing.

The rate match FIFO inserts or deletes as many /I2/ or /C2/ (first two bytes) as necessary to perform the rate match operation.

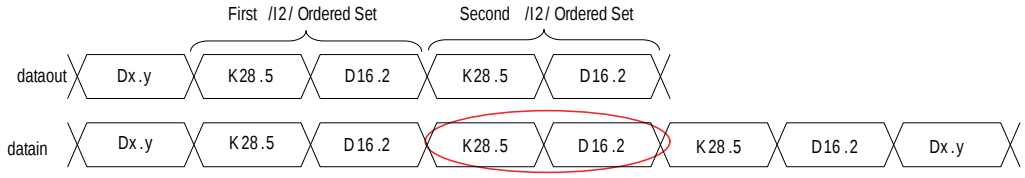
The following figure shows the rate match deletion operation where three symbols must be deleted. Because the rate match FIFO can only delete /I2/ ordered sets, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 2-31: Rate Match FIFO Deletion



The following figure shows an example of rate match FIFO insertion in the case where one symbol must be inserted. Because the rate match FIFO can only insert /I2/ ordered sets, it inserts one /I2/ ordered set (two symbols inserted).

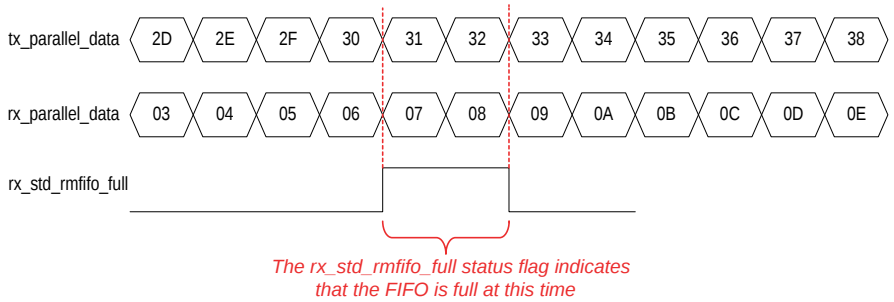
Figure 2-32: Rate Match FIFO Insertion



`rx_std_rmfifo_full` and `rx_std_rmfifo_empty` are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

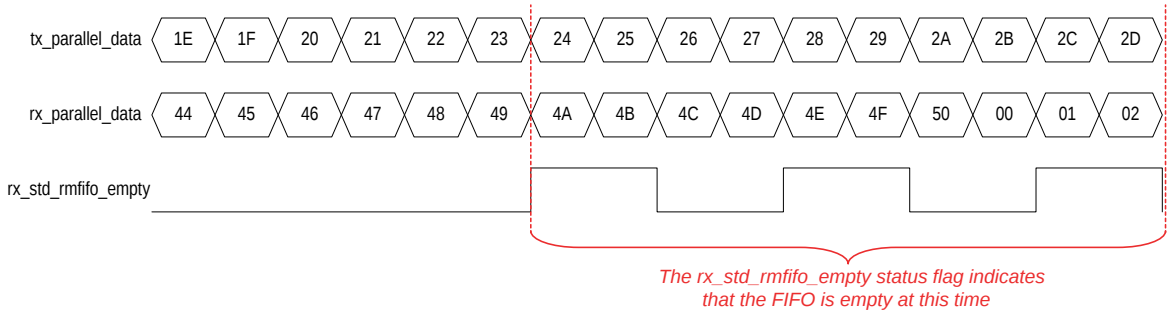
The rate match FIFO does not delete code groups to overcome a FIFO full condition. It asserts the `rx_std_rmfifo_full` flag for at least two recovered clock cycles to indicate rate match FIFO full. The following figure shows the rate match FIFO full condition when the write pointer is faster than the read pointer.

Figure 2-33: Rate Match FIFO Full Condition



The rate match FIFO does not insert code groups to overcome the FIFO empty condition. It asserts the `rx_std_rmfifo_empty` flag for at least two recovered clock cycles to indicate that the rate match FIFO is empty. The following figure shows the rate match FIFO empty condition when the read pointer is faster than the write pointer.

Figure 2-34: Rate Match FIFO Empty Condition



In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Related Information

Rate Match FIFO on page 5-41

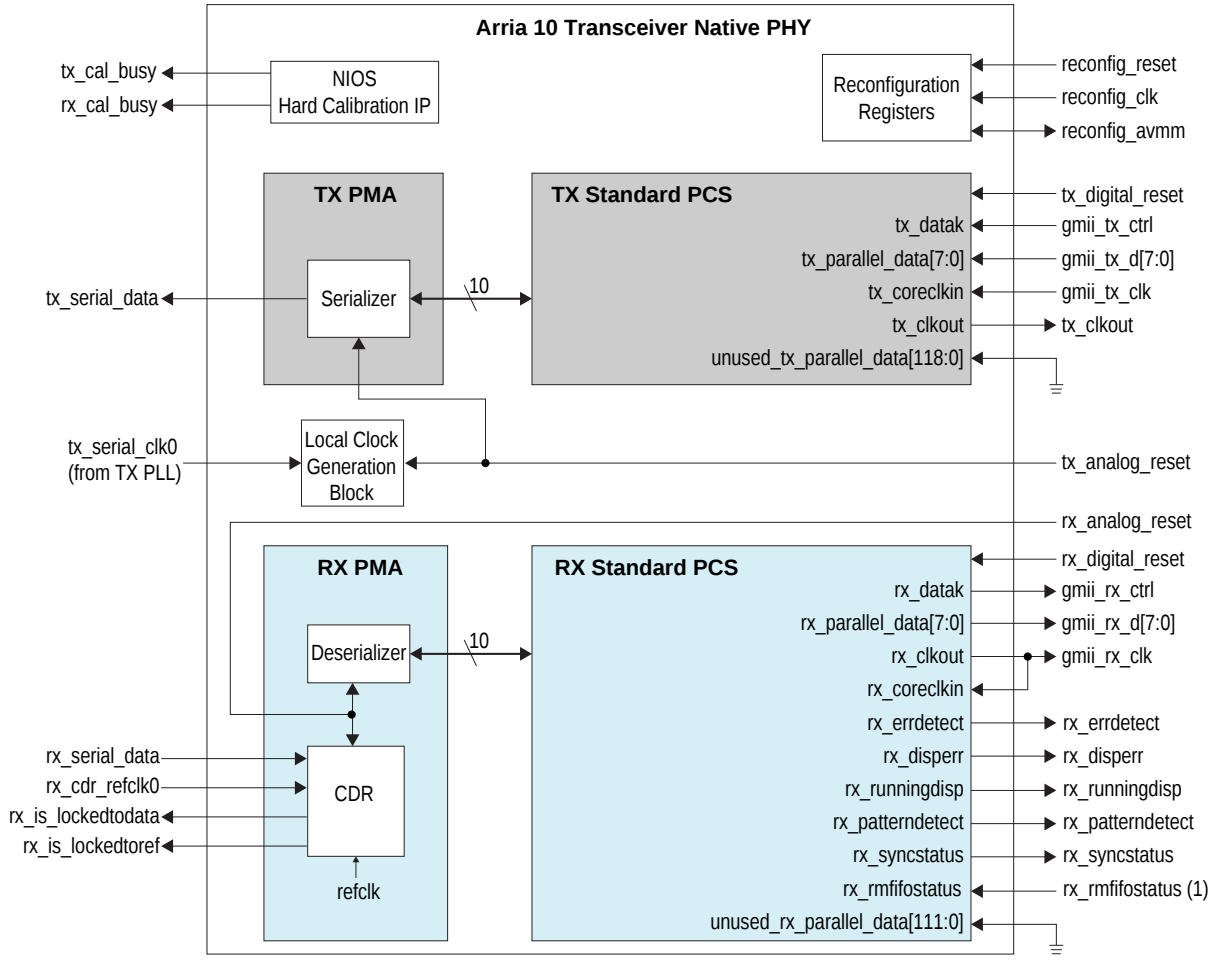
How to Implement GbE, GbE 1588 in Arria 10 Transceivers

Before you begin

You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing the GbE protocol.

1. Open the MegaWizard Plug-In Manager and select the PHY IP.
   Refer to **Select and Instantiate PHY IP** on page 2-2.
2. Select **GbE** or **GbE 1588** from the **Transceiver configuration rules** list located under **Datapath Options** depending on which protocol you are implementing.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for GbE and GbE with 1588** on page 2-88 as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the setting to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 2-35: Signals and Ports for Native PHY IP Configured for GbE, GbE with 1588

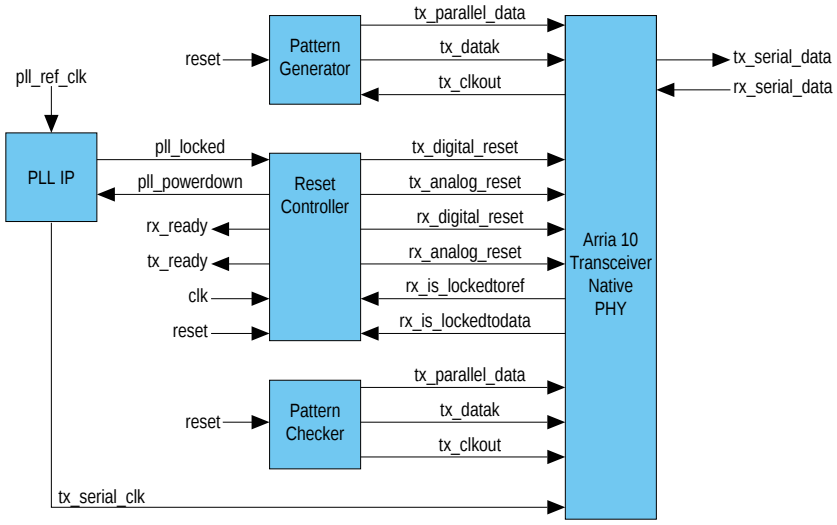


Note:
1. rx_rmfifostatus is not available in the GbE with 1588 configuration.

5. Instantiate and configure your PLL.
6. Instantiate a transceiver reset controller.

   You can use your own reset controller or use the Native PHY Reset Controller IP.
7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in **Figure 2-36** to connect the ports.

Figure 2-36: Connection Guidelines for a GbE/GbE with 1588 PHY Design



8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Standard PCS Architecture** on page 5-31
  For more information about Standard PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Standard PCS and PMA Ports** on page 2-52
  Port definitions for the Transceiver Native PHY Standard Datapath

Native PHY IP Parameter Settings for GbE and GbE with 1588

Table 2-68: General and Datapath Options

The first two sections of the MegaWizard Plug-In Manager for the Native PHY IP provide a list of general and datapath options to customize the transceiver.

Parameter	Value
Device speed grade	**fastest**

Parameter	Value
Message level for rule violations	error message
Transceiver configuration rules	• GbE (for GbE) • GbE 1588 (for GbE with 1588)
Transceiver mode	TX/RX Duplex TX Simplex RX Simplex
Number of data channels	1 to 96
Data rate	1250 Mbps
Enable reconfiguration between Standard and Enhanced PCSs	On/Off
Enable simplified data interface	On/Off

Table 2-69: TX PMA Parameters

Parameter	Value
TX channel bonding mode	Not bonded
TX local clock division factor	1, 2, 4, 8
Number of TX PLL clock inputs per channel	1, 2, 4, 8
Initial TX PLL clock input selection	0
Enable tx_pma_clkout port	On/Off
Enable tx_pma_div_clkout port	On/Off
tx_pma_div_clkout division factor	Disabled, 1, 2, 33, 40, 66
Enable tx_pma_elecidle port	On/Off
Enable tx_pma_qpipullup port (QPI)	On/Off
Enable tx_pma_qpipulldn port (QPI)	On/Off
Enable tx_pma_txdetectrx port (QPI)	On/Off
Enable tx_pma_rxfound port (QPI)	On/Off

Table 2-70: RX PMA Parameters

Parameter	Value
Number of CDR reference Clocks	1 to 5
Selected CDR reference clock	0 to 4

Parameter	Value
Selected CDR reference clock frequency	Select legal range defined by the Quartus II software
PPM detector threshold	**62.5**, **100**, **125**, **200**, **250**, **300**, **500**, **1000**
Decision feedback equalization mode	**Disabled**
Enable rx_pma_clkout port	**On/Off**
Enable rx_pma_div_clkout port	**On/Off**
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **50**, **66**
Enable rx_pma_clkslip port	**On/Off**
Enable rx_pma_qpipulldn port (QPI)	**On/Off**
Enable rx_is_lockedtodata port	**On/Off**
Enable rx_is_lockedtoref port	**On/Off**
Enable rx_set_locktodata and rx_set_locktoref ports	**On/Off**
Enable rx_seriallpbken port	**On/Off**
Enable PRBS verifier control and status ports	**On/Off**

Table 2-71: Standard PCS Parameters

Parameters	Value
Standard PCS / PMA interface width	**10**
FPGA fabric / Standard TX PCS interface width	**10**
FPGA fabric / Standard RX PCS interface width	**10**
TX FIFO mode	• **low latency** (for GbE) • **register_fifo** (for GbE with 1588)
RX FIFO Mode	• **low latency** (for GbE) • **register_fifo** (for GbE with 1588)
Enable Standard PCS low latency mode	**Off**
Enable tx_std_pcfifo_full port	**On/Off**
Enable tx_std_pcfifo_empty port	**On/Off**
Enable rx_std_pcfifo_full	**On/Off**
Enable rx_std_pcfifo_empty port	**On/Off**
TX byte serializer mode	**Disabled**, **Serialize x2**
RX byte deserializer mode	**Disabled**, **Deserialize x2**
Enable TX 8B/10B encoder	**On/Off**

Parameters	Value
Enable TX 8B/10B disparity control	**On/Off**
Enable RX 8B/10B decoder	**On/Off**
RX rate match FIFO mode	• **gige** (for GbE) • **disabled** (for GbE with 1588)
RX rate match insert / delete -ve pattern (hex)	• `0x000ab683` for (GbE) • `0x00000000` (for GbE with 1588)
RX rate match insert / delete +ve pattern (hex)	• `0x000a257c` for (GbE) • `0x00000000` (for GbE with 1588)
Enable rx_std_rmfifo_full port	**On/Off**
Enable rx_std_rmfifo_empty port	**On/Off**
PCI Express Gen3 rate match FIFO mode	**Bypass**
Enable TX bit slip	**Off**
Enable tx_std_bitslipboundarysel port	**On/Off**
RX word aligner mode	**Synchronous state machine**
RX word aligner pattern length	**7, 10**
RX word aligner pattern (hex)	`0x000000000000007c`, `0x000000000000017c`
Number of word alignment patterns to achieve sync	**3**
Number of invalid data words to lose sync	**3**
Number of valid data words to decrement error count	**3**
Enable rx_std_wa_patternalign port	**Off**
Enable rx_std_wa_a1a2size port	**Off**
Enable rx_std_bitslipboundarysel port	**Off**
Enable rx_bitslip port	**Off**
Enable TX bit reversal	**Off**
Enable TX byte reversal	**Off**
Enable TX polarity inversion	**On/Off**
Enable tx_polinv port	**On/Off**
Enable RX bit reversal	**Off**
Enable rx_std_bitrev_ena port	**Off**
Enable RX byte reversal	**Off**
Enable rx_std_byterev_ena port	**Off**

Parameters	Value
Enable RX polarity inversion	**On/Off**
Enable rx_polinv port	**On/Off**
Enable rx_std_signaldetect port	**On/Off**
All options under **PCIe Ports**	**Off**

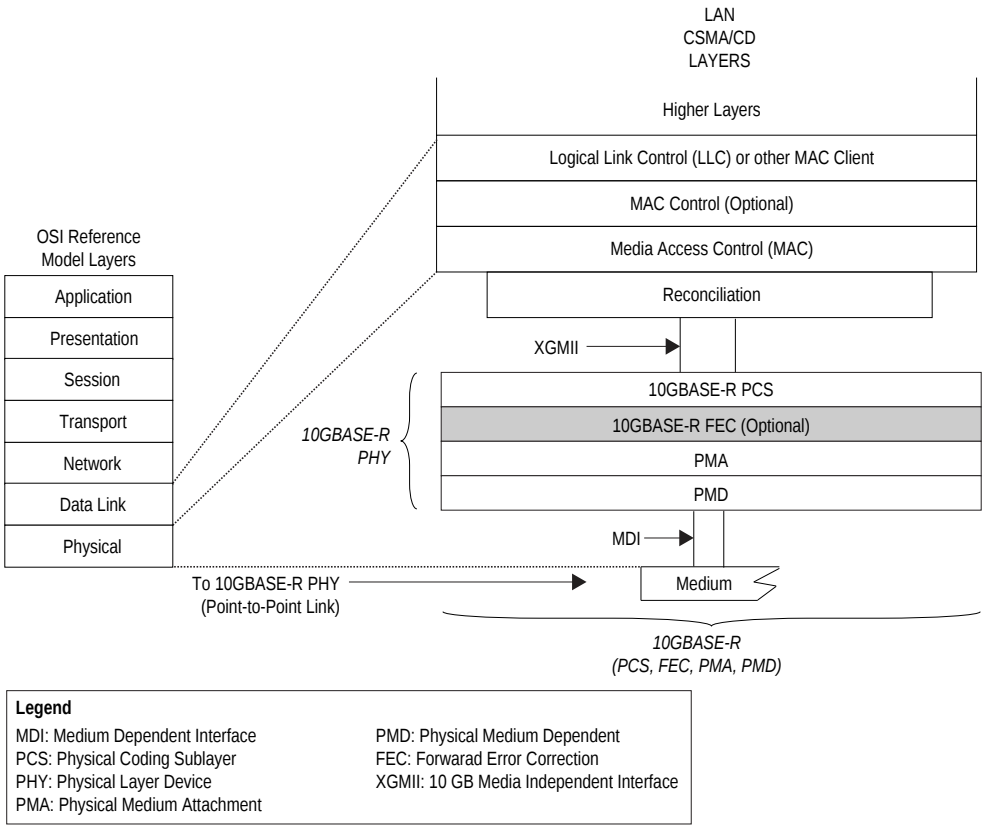
10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC

Arria 10 transceivers can implement the 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with FEC protocols using the 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC Transceiver Configuration Rules, respectively.

10GBASE-R is a specific physical layer implementation of the 10 Gigabit Ethernet link defined in clause 49 of the IEEE 802.3-2008 specification.

The 10GBASE-R protocol is used in optical module LAN applications such as optical routers, servers, and switches. It uses the XGMII interface to connect to the IEEE802.3 media access control (MAC) and reconciliation sub-layers (RS). It delivers serialized data at a line rate of 10.3125 gigabits per second (Gbps). Each channel operates independently in a multi-channel implementation.

Figure 2-37: 10GBASE-R PHY Connection to IEEE802.3 MAC and RS



You can configure the transceivers for 10GBASE-R functionality by using the Native PHY IP to implement the PHY layer of the 10GBASE-R Transceiver Configuration Rule. The Native PHY IP must be connected to a third-party PHY MAC layer to create a complete 10GBASE-R design.

10GBASE-R with 1588

Arria 10 transceivers can implement the 10GBASE-R 1588 protocol using the 10GBASE-R 1588 Transceiver Configuration Rules.

The IEEE 1588 Precision Time Protocol is used for precise synchronization of clocks in distributed systems in telecommunications, power generation and distribution, industrial automation, robotics, data acquisition, test, and measurement. The protocol is applicable to systems communicating by local area networks including, but not limited to, Ethernet. The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock.

The 10GBASE-R 1588 protocol mode supports a lane rate of 10.3125 Gbps. The PCS sublayer interfaces with the MAC through the gigabit medium independent interface (GMII).

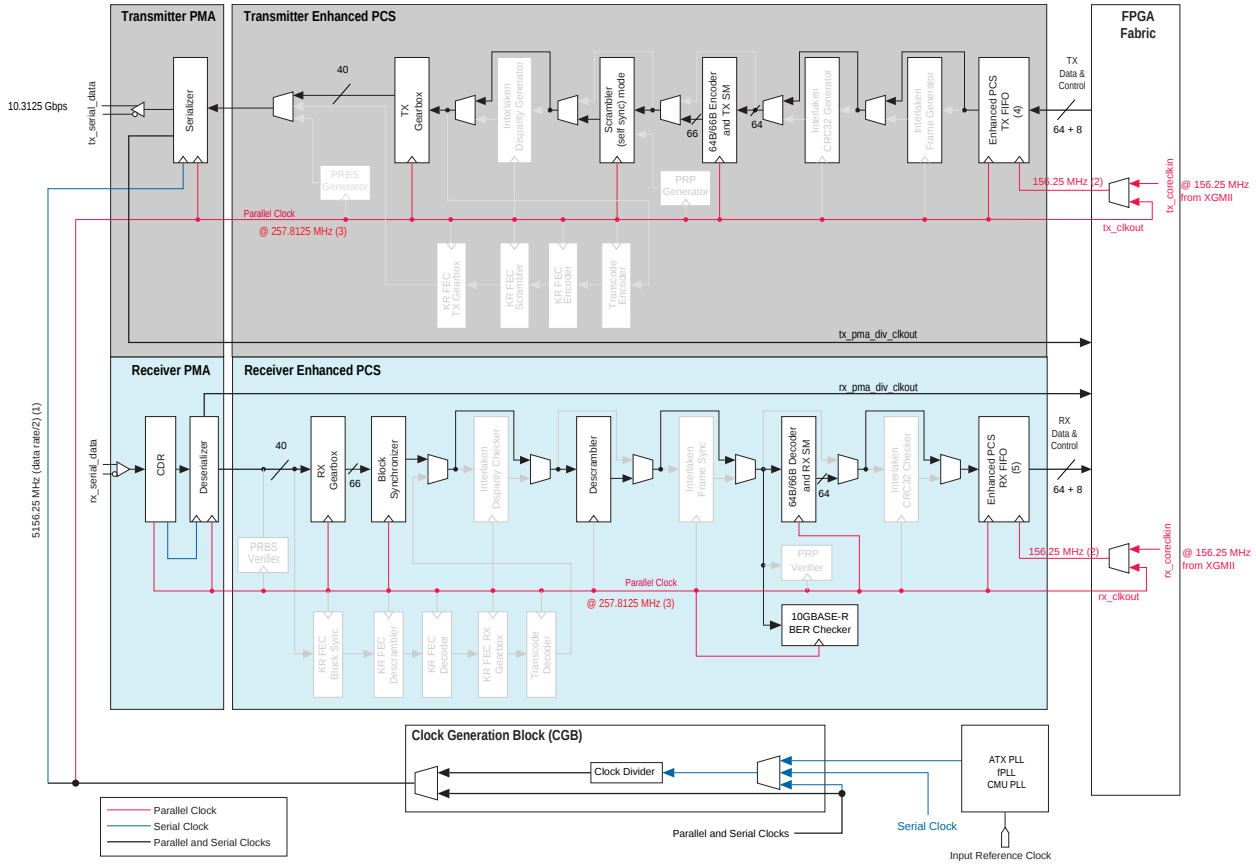
Use the Native PHY IP to implement the PHY Layer of the 10GBASE-R 1588 protocol. The Native PHY IP must be connected to a third-party PHY MAC layer to create a complete 10GBASE-R 1588 design.

10GBASE-R with KR FEC

The 10GBASE-R with KR FEC protocol is a KR FEC sublayer placed between the PCS and PMA sublayers of the 10GBASE-R physical layer.

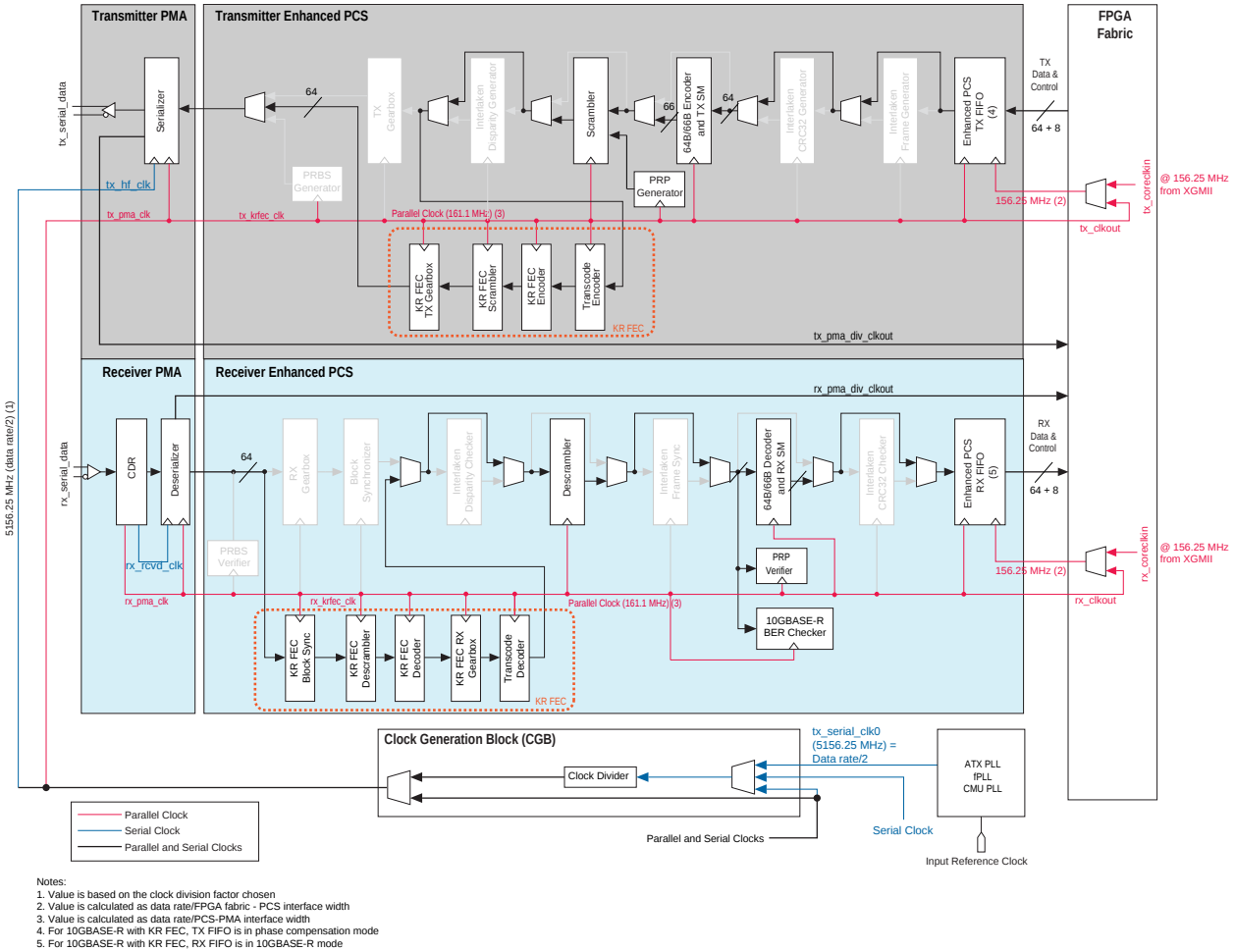
The Open Systems Interconnection (OSI) reference model for this protocol is shown in the following figure. The KR FEC sublayer increases the bit error rate (BER) performance of 10GBASE-R by providing additional link margin to account for variation in manufacturing and environmental conditions.

Figure 2-38: Transceiver Channel Datapath and Clocking for 10GBASE-R, 10GBASE-R with 1588



Notes:
1. Value based on the the clock division factor chosen.
2. Value calculated as data rate / FPGA fabric-PCS interface width.
3. Value calculated as data rate / PCS-PMA interface width.
4. This block is in Phase Compensation mode for the 10GBASE-R configuration and register mode for the 10GBASE-R with 1588 configuration.
5. This block is in 10GBASE-R mode for the 10GBASE-R configuration and register mode for the 10GBASE-R with 1588 configuration.

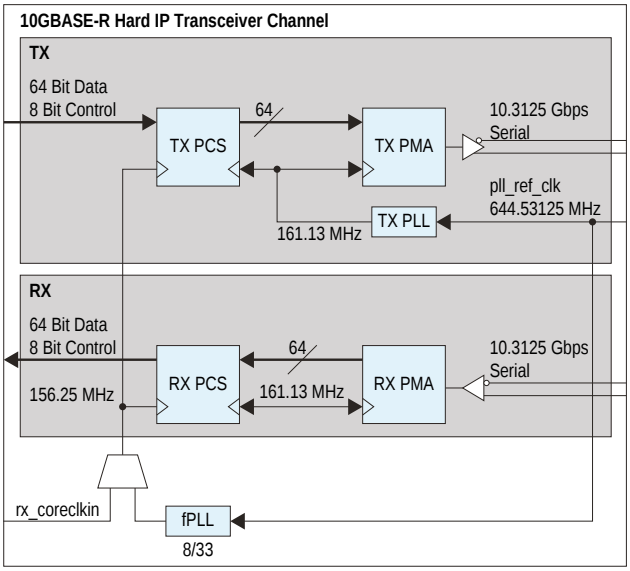
Figure 2-39: Transceiver Channel Datapath and Clocking for 10GBASE-R with KR FEC



Notes:
1. Value is based on the clock division factor chosen
2. Value is calculated as data rate/FPGA fabric - PCS interface width
3. Value is calculated as data rate/PCS-PMA interface width
4. For 10GBASE-R with KR FEC, TX FIFO is in phase compensation mode
5. For 10GBASE-R with KR FEC, RX FIFO is in 10GBASE-R mode

The CMU PLL or the ATX PLLs generate the TX high-speed serial clock. The following figure shows a way of clock generation and distribution in Arria 10 devices for 10GBASE-R with KR FEC protocol.

Figure 2-40: Clock Generation and Distribution for 10GBASE-R with KR FEC

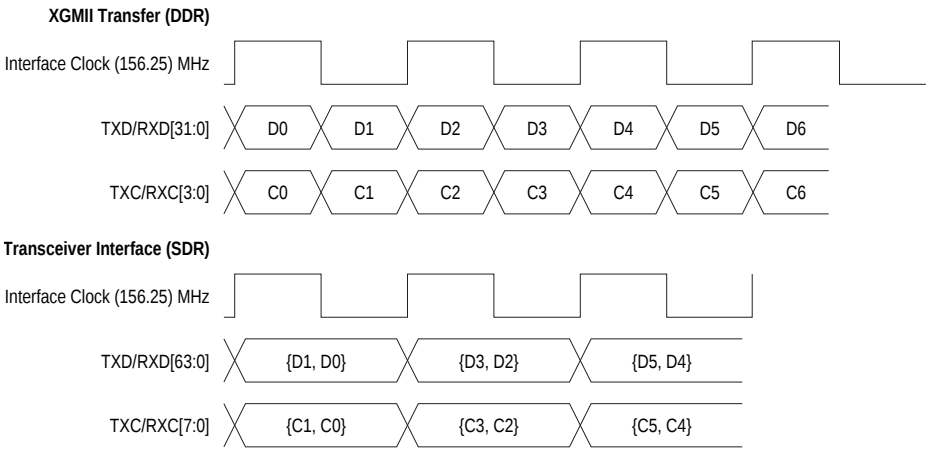


Clocking in 10GBASE-R and 10GBASE-R with 1588

The XGMII interface defines the 32-bit data and 4-bit wide control character clocked between the MAC/RS and the PCS at both the positive and negative edge (double data rate – DDR) of the 156.25 MHz interface clock.

The transceivers do not support the XGMII interface to the MAC/RS as defined in the IEEE 802.3-2008 specification. Instead, they support a 64-bit data and 8-bit control single data rate (SDR) interface between the MAC/RS and the PCS.

Figure 2-41: XGMII Interface (DDR) and Transceiver Interface (SDR) for 10GBASE-R Configurations



Note: Clause 46 of the IEEE 802.3-2008 specification defines the XGMII interface between the 10GBASE-R PCS and the Ethernet MAC/RS.

The transceiver PLL supports a reference clock frequency of 322.265625 MHz and 644.53125 MHz for the 10GBASE-R and 10GBASE-R with 1588 Transceiver Configuration Rules.

For 10GBASE-R, to ensure proper functioning of the PCS, the maximum ppm difference between the TX PLL reference clock and XGMII interface clock on the transmit side is 0 ppm. The RX FIFO can compensate ±100 ppm between the RX low-speed parallel clock and XGMII interface clock on the receive side.

Note: Channel bonding should be disabled when using the 10GBASE-R and 10GBASE-R with 1588 Transceiver Configuration Rules.

TX FIFO and RX FIFO

In 10GBASE-R configuration, the TX FIFO behaves as a phase compensation FIFO and the RX FIFO behaves as a clock compensation FIFO.

In 10GBASE-R with 1588 configuration, both the TX FIFO and the RX FIFO are used in register mode.

In 10GBASE-R with KR FEC configuration, the TX FIFO is used in phase compensation mode and the RX FIFO behaves as a clock compensation FIFO.

Related Information

Arria 10 Enhanced PCS Architecture on page 5-14
For more information about the Enhanced PCS Architecture

How to Implement 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC in Arria 10 Transceivers

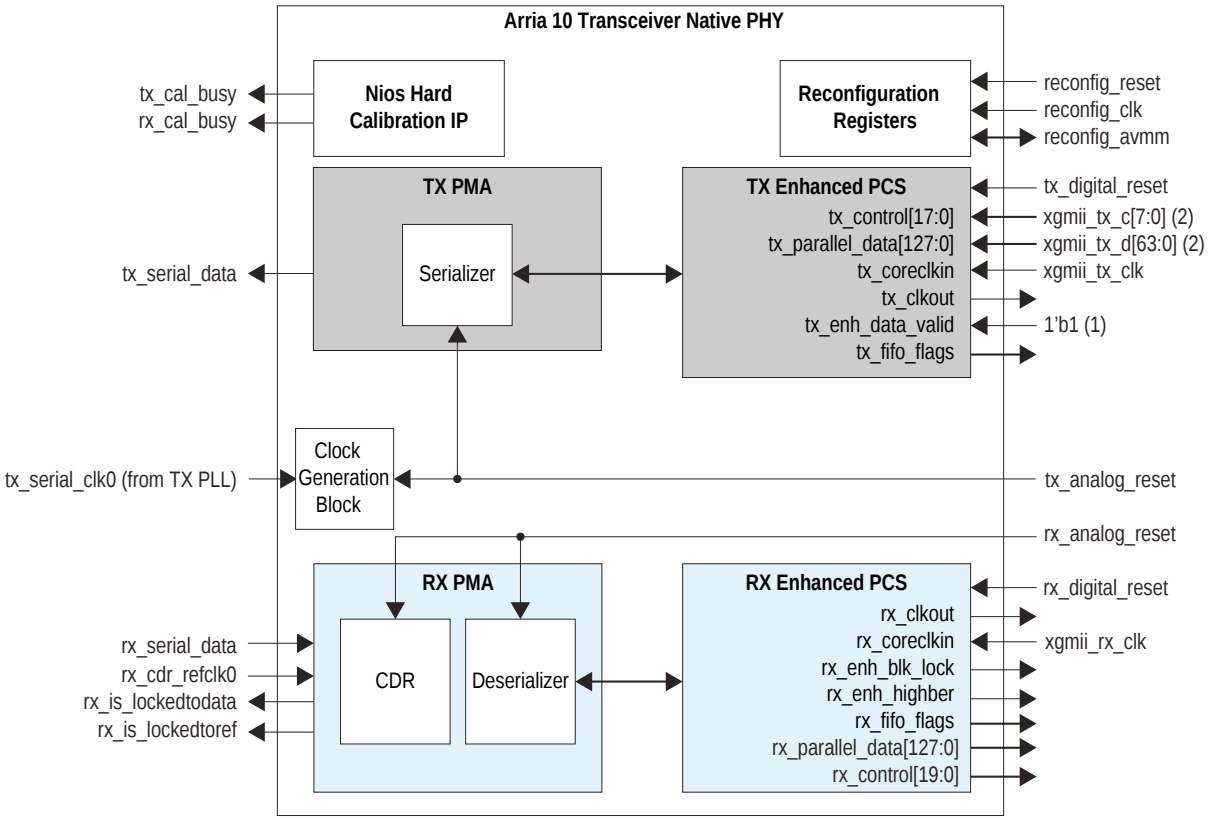
Before you begin

You should be familiar with the 10GBASE-R and PMA architecture, PLL architecture, and the reset controller before implementing the 10GBASE-R, 10GBASE-R with 1588, or 10GBASE-R with KR FEC Transceiver Configuration Rules.

You must design your own MAC and other layers in the FPGA to implement the 10GBASE-R, 10GBASE-R with 1588, or 10GBASE-R with KR FEC Transceiver Configuration Rule using the Native PHY IP.

1. Open the MegaWizard Plug-In Manager and select the **Arria 10 Transceiver Native PHY IP**. Refer to **Select and Instantiate PHY IP** on page 2-2 for more details.
2. Select **10GBASE-R**, **10GBASE-R 1588**, or **10GBASE-R with KR FEC** from the **Transceiver configuration rule** list located under **Datapath Options** depending on which protocol you are implementing.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC** as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 2-42: Signals and Ports of Native PHY IP for the 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC



Notes:
1. For 10GBASE-R with 1588 configurations, this signal is user-controlled.
2. For 10GBASE-R with 1588 configurations, this signal is connected from the output of TX FIFO in the FPGA fabric.

5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Arria 10 Transceiver Native PHY Reset Controller IP.
7. Connect the Arria 10 Transceiver Native PHY to the PLL IP and the reset controller.

Figure 2-43: Connection Guidelines for a 10GBASE-R or 10GBASE-R with KR FEC PHY Design

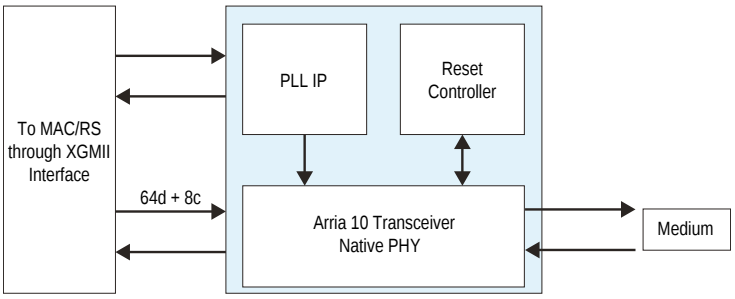
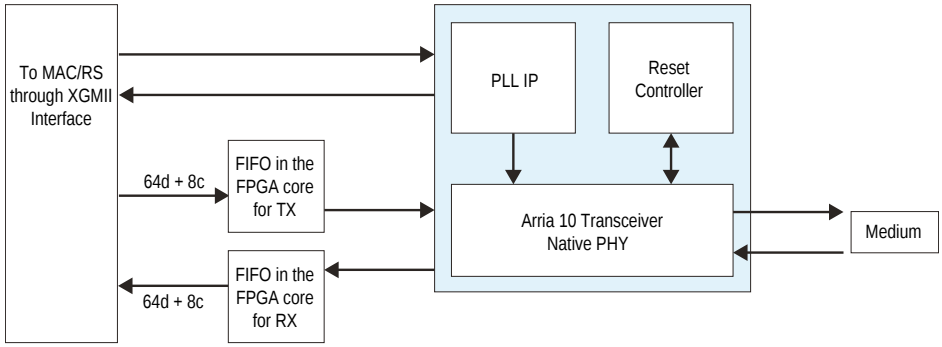


Figure 2-44: Connection Guidelines for a 10GBASE-R with 1588 PHY Design



8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Enhanced PCS Architecture** on page 5-14
  For more information about Enhanced PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Enhanced PCS and PMA Ports** on page 2-37
  For detailed information about the available ports in the 10GBASE-R 1588 protocol.

Native PHY IP Parameter Settings for 10GBASE-R, 10GBASE-R with 1588, and 10GBASE-R with KR FEC

Table 2-72: General and Datapath Parameters

The first two sections of the MegaWizard Plug-In Manager for the Transceiver Native PHY provide a list of general and datapath options to customize the transceiver.

Parameter	Range
Device speed grade	**fastest**
Message level for rule violations	**error, warning**
Transceiver Configuration Rule	• **10GBASE-R** • **10GBASE-R 1588** • **10GBASE-R with KR FEC**
Transceiver mode	**TX / RX Duplex, TX Simplex, RX Simplex**

Parameter	Range
Number of data channels	**1** to **96**
Data rate	**10312.5 Mbps**
Enable reconfiguration between Standard and Enhanced PCS	**On / Off**
Enable simplified data interface	**Off**

Table 2-73: TX PMA Parameters

Parameter	Range
TX channel bonding mode	**Non-bonded**
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0**

Table 2-74: RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	**322.265625** and **644.53125** MHz
PPM detector threshold	**62.5**, **100**,**125**, **200**, **250**, **300**, **500**, **1000**
Decision feedback equalization mode	**disabled**

Table 2-75: Enhanced PCS Parameters

Parameter	Range
Enhanced PCS/PMA interface width	**32**, **40**, **64** **Note:** 10GBASE-R with KR FEC allows **64** only.
FPGA fabric/Enhanced PCS interface width	**66**
Enable RX/TX FIFO double-width mode	**Off**

Parameter	Range
TX FIFO mode	• **Phase Compensation** (10GBASE-R and 10GBASE-R with KR FEC) • **Register** (10GBASE-R with 1588)
TX FIFO partially full threshold	11
TX FIFO partially empty threshold	2
RX FIFO mode	• **10GBASE-R** (10GBASE-R and 10GBASE-R with KR FEC) • **Register** (10GBASE-R with 1588)
RX FIFO partially full threshold	23
RX FIFO partially empty threshold	2

Table 2-76: 64B/66B Encoder and Decoder Parameters

Parameter	Range
Enable TX 64B/66B encoder	On
Enable RX 64B/66B decoder	On
Enable TX sync header error insertion	On / Off

Table 2-77: Scrambler and Descrambler Parameters

Parameter	Range
Enable TX scrambler (10GBASE-R / Interlaken)	On
TX scrambler seed (10GBASE-R / Interlaken)	0x03ffffffffffffff
Enable RX descrambler (10GBASE-R / Interlaken)	On

Table 2-78: Block Sync Parameters

Parameter	Range
Enable RX block synchronizer	On

Table 2-79: Gearbox Parameters

Parameter	Range
Enable TX data polarity inversion	On / Off
Enable RX data polarity inversion	On / Off

Table 2-80: Dynamic Reconfiguration Parameters

Parameter	Range
Enable dynamic reconfiguration	On / Off
Share reconfiguration interface	On / Off
Enable embedded JTAG AVMM master	On / Off

Table 2-81: Configuration Files Parameters

Parameter	Range
Configuration file prefix	—
Generate SystemVerilog package file	On / Off
Generate C header file	On / Off
Generate MIF (Memory Initialization File)	On / Off

Table 2-82: Generation Options Parameters

Parameter	Range
Generate parameter documentation file	On / Off

Native PHY IP Ports for 10GBASE-R and 10GBASE-R with 1588 Transceiver Configurations

Figure 2-45: High BER

This figure shows the `rx_enh_highber` status signal going high when there are errors on the `rx_parallel_data` output.

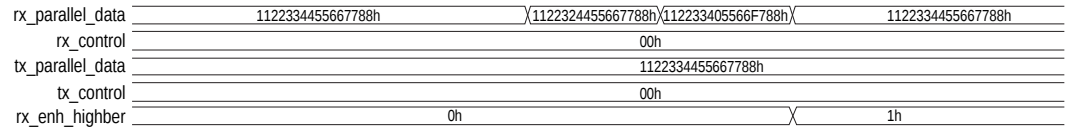
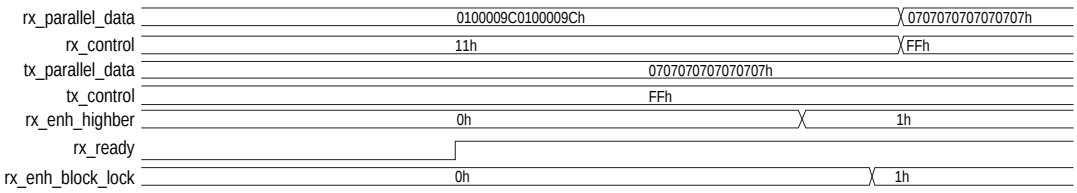


Figure 2-46: Block Lock Assertion

This figure shows the assertion on `rx_enh_blk_lock` signal when the Receiver detects the block delineation.



The following figures show Idle insertion and deletion.

Figure 2-47: IDLE Word Insertion

This figure shows the insertion of IDLE words in the receiver data stream.

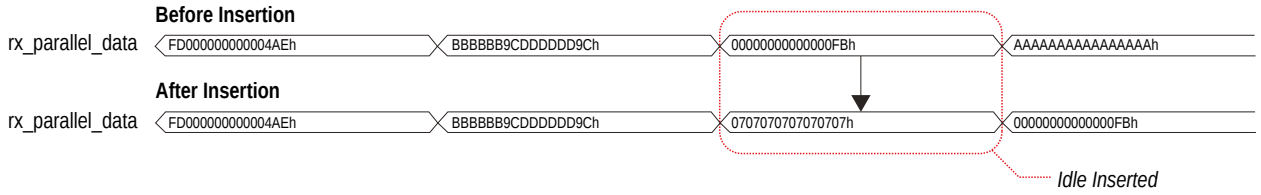


Figure 2-48: IDLE Word Deletion

This figure shows the deletion of IDLE words from the receiver data stream.

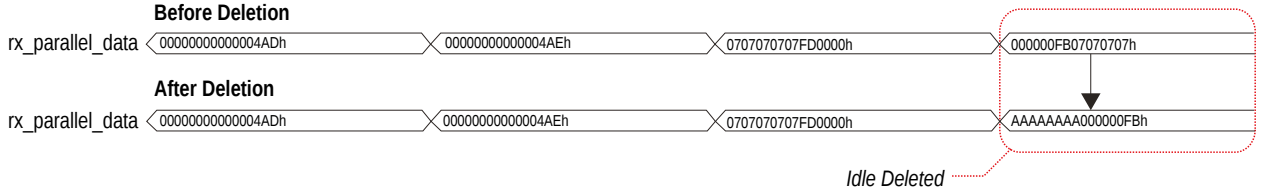
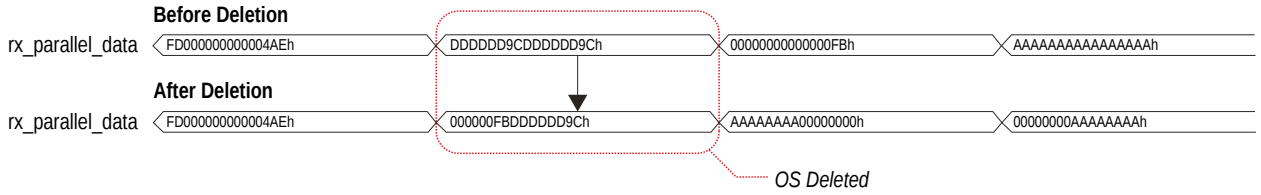


Figure 2-49: OS Word Deletion

This figure shows the deletion of Ordered set word in the receiver data stream.



10GBASE-KR PHY IP with FEC Option

The Ethernet standard comprises many different PHY standards with variations in signal transmission medium and data rates. The 1G/10GbE and 10GBASE-KR PHY IP Core enables Ethernet connectivity at 1 Gbps and 10 Gbps over backplanes. The 10GBASE-KR PHY IP is also known as the Backplane Ethernet PHY IP. It includes link training and auto negotiation to support the IEEE Backplane Ethernet standard.

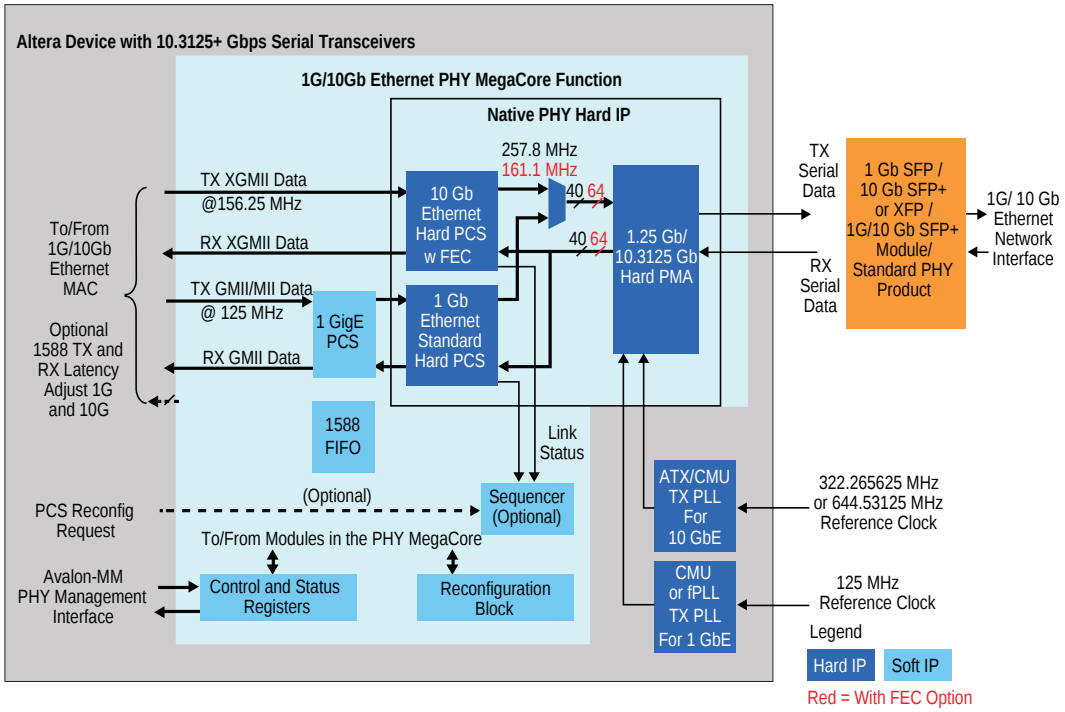
The 10GBASE-KR Ethernet PHY IP MegaCore® function supports the following features of Ethernet standards:

- 10GBASE-KR Ethernet protocol with link training as defined in *Clause 72* of the *IEEE 802.3 2008 Standard*. In addition to the link-partner TX tuning as defined in *Clause 72*, this PHY also automatically configures the local device RX interface for the lowest bit error rate (BER).
- Auto negotiation for backplane Ethernet as defined in *Clause 73* of the *IEEE 802.3 2008 Standard*. The 10GBASE-KR Ethernet PHY MegaCore Function can auto negotiate between 1000BASE-X, 1000BASE-KR , and 1000BASE-KRwith FEC.
- Gigabit Media Independent Interface (GMII) to connect PHY with media access control (MAC) as defined in *Clause 35* of the *IEEE 802.3 2008 Standard*
- Auto-negotiation as defined in*Clause 37* of the *IEEE 802.3 2008 Standard*
- Gigabit Ethernet protocol as defined in*Clause 49* of the *IEEE 802.3 2008 Standard*
- XGMII to provide simple and inexpensive interconnection between the MAC and the PHY as defined in *Clause 46* of the *IEEE 802.3 2008 Standard*
- Forward Error correction(FEC) as defined in *Clause 74* of the *IEEE 802.3 2008 Standard*
- Precision time protocol (PTP) as defined in the *IEEE 1588 Standard*
- 10M/100Mbps MII to connect physical media with the MAC as defined in *Clause 22* of the *IEEE 802.1 2008 Standard*

Using the Backplane Ethernet PHY MegaCore function, you can implement the 1GbE protocol using the Standard PCS and 10GbE protocol using Enhanced PCS and PMA. You can switch dynamically between the 1G and 10G data rates using dynamic reconfiguration to reprogram the transceivers. Or, you can use the speed detection option to automatically switch data rates based on received data.

The following figure shows the top-level modules of the 1G/10GbE PHY IP. As show in the following figure, the 1G/10 Gbps Ethernet PHY connects to a separately instantiated MAC. The Enhanced PCS receives and transmits XGMII data. The Standard PCS receives and transmits GMII data. An Avalon® Memory-Mapped (Avalon-MM) slave interface provides access to PCS registers. The PMA receives and transmits serial data.

Figure 2-50: Top Level Modules of the 1G/10GbE PHY MegaCore Function



An Avalon Memory-Mapped (Avalon-MM) slave interface provides access to the 1G/10GbE PHY IP Core registers. These registers control the functions of the blocks shown in the above figure.

The Backplane Ethernet 10GBASE-KR PHY IP includes the following new modules to enable operation over a backplane:

- Link Training (LT)— The LT mechanism allows the 10GBASE-KR PHY to automatically configure the link-partner TX PMDs for the lowest Bit Error Rate (BER). LT is defined in Clause 72 of IEEE Std 802.3ap-2007.
- Auto negotiation (AN)—The 10GBASE-KR PHY IP can auto-negotiate between 1000BASE-KX (1GbE) and 10GBASE-KR (10GbE) PHY types. The AN function is mandatory for Backplane Ethernet. It is defined in Clause 73 of the IEEE Std 802.3ap-2007.
- Forward Error Correction—Forward Error Correction (FEC) function is an optional feature defined in *Clause 74* of *IEEE 802.3ap-2007*. It provides an error detection and correction mechanism allowing noisy channels to achieve the Ethernet-mandated Bit Error Rate (BER) of 10^{-12} .

Related Information

- **IEEE Std 802.3ap-2008 Standard**
- **Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems**

10GBASE-KR PHY Release Information

Table 2-83: 10GBASE-KR PHY Release Information

Item	Description
Version	13.1
Release Date	November 2013
Ordering Codes	IP-10GBASEKR PHY (primary)
Product ID	0106
Vendor ID	6AF7

10GBASE-KR PHY Performance and Resource Utilization

This topic provides performance and resource utilization for the IP.

The following table shows the typical expected resource utilization for selected configurations using the Quartus II software v13.1 for Arria 10 devices. The numbers of ALMs and logic registers are rounded up to the nearest 100.

Table 2-84: 10GBASE-KR PHY Performance and Resource Utilization

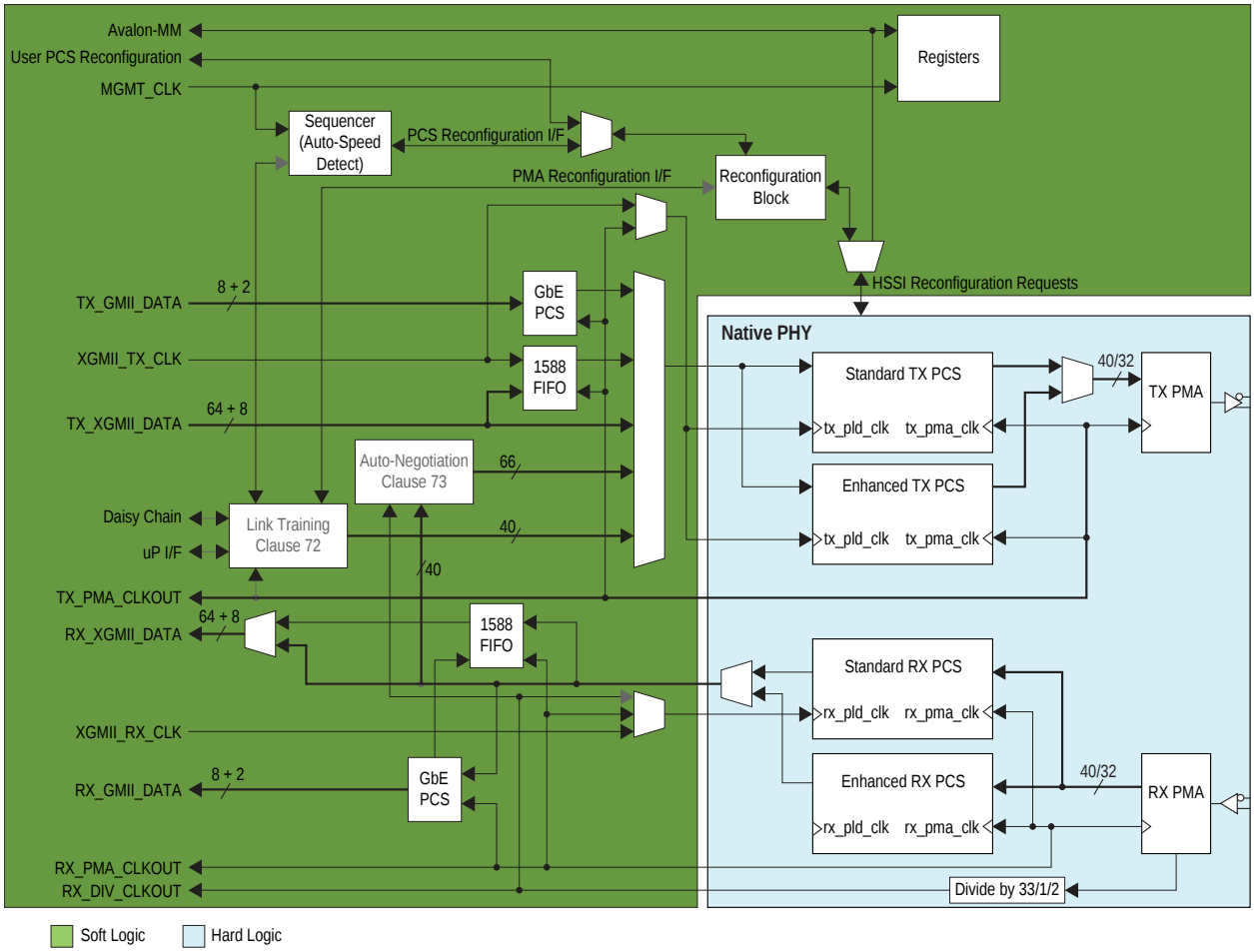
Variant	ALMs	ALUTs	Registers	M20K
10GBASE-KR PHY with 1588	4700	6600	6750	5
10GBASE-KR PHY	2400	3750	3100	1
10GBASE-KR PHY with FEC	2400	3750	3100	1

10GBASE-KR Functional Description

This topic provides high-level block diagram.

The following figure shows the 10GBASE-KR PHY IP and the supporting modules required for integration into your system.

Figure 2-51: 10GBASE-KR PHY IP Block Diagram



As this figure illustrates, the Backplane Ethernet 10GBASE-KR PHY IP core is built using the Native PHY. It includes the following modules:

Standard and Enhanced PCS Datapaths

The Standard PCS supports the 1G protocol. The Enhanced PCS supports 10GBASE-R. Refer to the Standard PCS and Enhanced PCS architecture chapters for more details on how these blocks support 1G, 10G protocols and FEC.

Sequencer

The Sequencer controls the start-up sequence of the PHY IP, including reset and power-on. It selects which PCS (1G or 10G) and PMA interface is active. The Sequencer interfaces to the reconfiguration block to request a reconfigurations to change from one data rate to the other data rate.

Gigabit Ethernet (GbE) PCS

The GbE PCS includes the GMII interface and *Clause 37* auto negotiation and SGMII functionality.

1588 FIFO

The 1588 FIFO has an XGMII-like interface for both input and outputs. The 1588 FIFO includes the latency adjust information that the 1588 logic in the MAC requires.

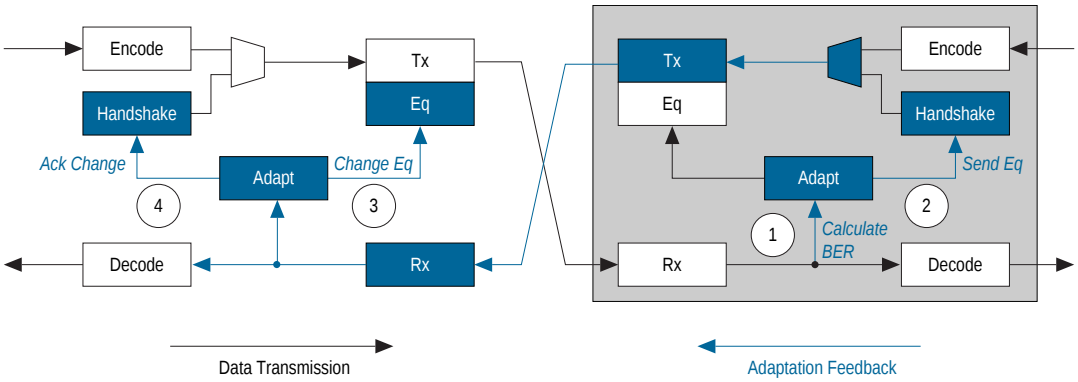
Link Training (LT), Clause 72

This module performs link training as defined in *Clause 72*. The module facilitates two features:

- Daisy-chain mode for non-standard link configurations where the TX and RX interfaces connect to different link partners instead of in a spoke and hub or switch topology.
- An embedded processor mode to override the state-machine-based training algorithm. This mode allows an embedded processor to establish link data rates instead of establishing the link using the state-machine-based training algorithm.

The following figure illustrates the link training process, where the link partners exchange equalization data.

Figure 2-52: TX Equalization for Link Partners



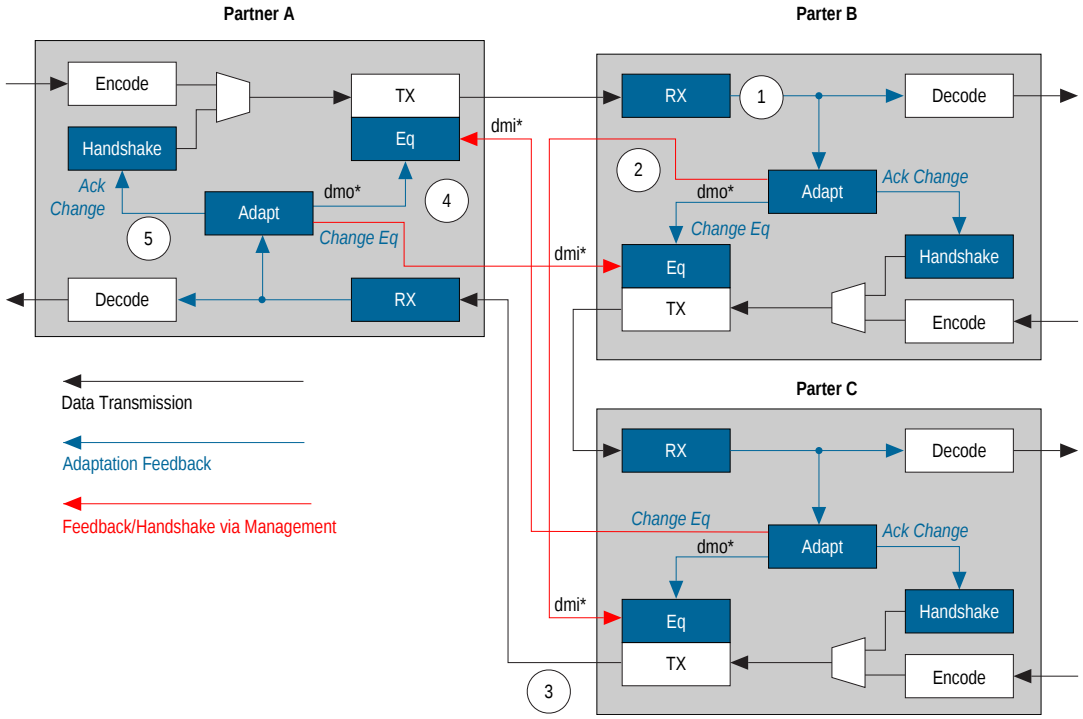
TX equalization includes the following steps which are identified in this figure.

1. The receiving link partner calculates the BER.
2. The receiving link partner transmits an update to the transmitting link partner TX equalization parameters to optimize the TX equalization settings
3. The transmitting partner updates its TX equalization settings.
4. The transmitting partner acknowledges the change.

This process is performed first for the V_{OD}, then the pre-emphasis, the first post-tap, and then pre-emphasis pre-tap.

The optional backplane daisy-chain topology can replace the spoke or hub switch topology. The following illustration highlights the steps required for TX Equalization for Daisy Chain Mode.

Figure 2-53: TX Equalization in Daisy-Chain Mode



Data transmission proceeds clockwise from link partner A, to B, to C. TX equalization includes the following steps which are identified in the figure :

1. The receiving partner B calculates the BER for data received from transmitting partner A.
2. The receiving partner B sends updates for TX link partner C.
3. The receiving link partner C transmits an update to the transmitting link partner A.
4. Transmit partner A updates its equalization settings.
5. Transmit partner A acknowledges the change.

This procedure is repeated for the other two link partners.

Auto Negotiation

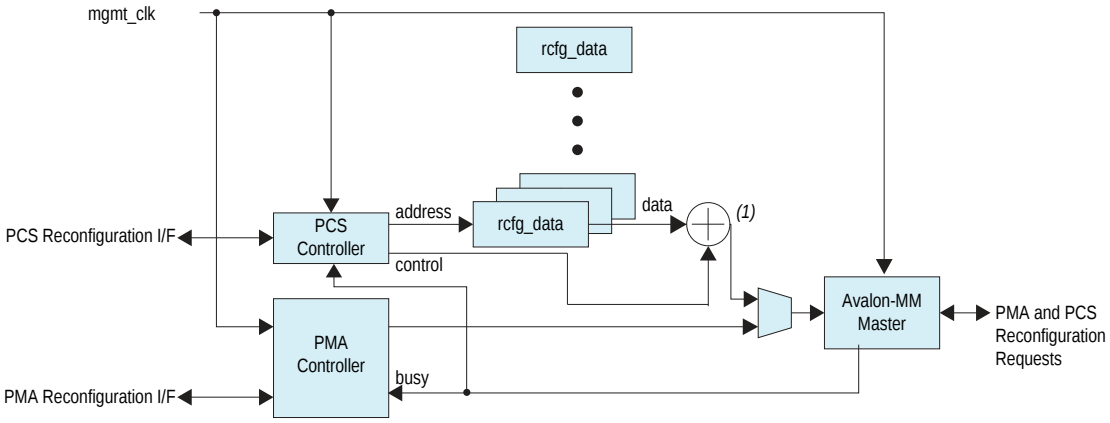
The Auto Negotiation module in the 10GBASE-KR PHY IP implements *Clause 73* of the *IEEE 802.3 Standard 2008*. This module currently supports auto negotiation between 1GbE and 10GBASE-R data rates. Auto negotiation with XAUI is not supported. Auto negotiation runs at power-up or if the sequencer module is reset.

Reconfiguration Block

The Reconfiguration Block performs the Avalon-MM writes to the PHY for both PCS and PMA reconfiguration. The following figure shows the reconfiguration block details. The Avalon-MM master accepts requests from the PMA or PCS controller. It performs the Read-Modify-Write or Write commands using the Avalon-MM interface. The PCS controller receives rate change requests from the Sequencer and translates them to a series of Read-Modify-Write or Write commands to the PMA and PCS.

Eight compile-time configuration modes are supported. These include the enumerated combination of reference clock (644 MHz or 322 MHz), including the 1588 mode, and the FEC sublayer

Figure 2-54: Reconfiguration Block Details



Note: Based on the control signal, the data is streamed to the AVMM master.

Related Information

- **Arria 10 Enhanced PCS Architecture** on page 5-14
- **Arria 10 Standard PCS Architecture** on page 5-31

Parameterizing the 10GBASE-KR PHY

The Arria 10 1G/10GbE and 10GBASE-KR PHY IP core allows you to select either the **Backplane-KR** or **1Gb/10Gb Ethernet** variant. When you select the **Backplane-KR** variant, the **Link Training (LT)** and **Auto Negotiation (AN)** tabs appear. The **1Gb/10Gb Ethernet** variant (1G/10GbE) does not implement the LT and AN functions.

Complete the following steps to parameterize the 10GBASE-KR PHY IP core in the MegaWizard Plug-In Manager:

1. For **Which device family will you be using?**, select Arria 10 from the list.
2. Click **Installed Plug-Ins** > **Interfaces** > **Ethernet**> **Arria 10 1G10GbE and 10BASE-KR PHY**.
3. Select **10GBASE-KR PHY** version from the **IP variant** list in MegaWizard Plug-In Manager.
4. Use the tabs on the MegaWizard Plug-In Manager to select the options required for the protocol.
5. Refer to the topics listed as Related Links to understand and specify 1G/10GbE parameters.
6. Click **Finish** to generate your parameterized 10GBASE-KR PHY IP Core.

Related Information

- **10GBASE-R Parameters** on page 2-111
- **10M/100M/1Gb Ethernet Parameters** on page 2-112
- **Speed Detection Parameters** on page 2-113
- **10GBASE-KR Auto-Negotiation Parameters** on page 2-114
- **1GbE Parameters**
- **10GBASE-KR Link Training Parameters**

General Options

The General Options allow you to specify options common to 1GbE and 10GbE modes.

Table 2-85: General Options Parameters

Parameter Name	Options	Description
Initial datapath	**10G , 1G**	Specifies the data rate need after reset or power up. If you select **1G** for the initial datapath, the automatic speed detection function is not available.
Enable internal PCS reconfiguration logic	**On/Off**	When you turn this option on, the core includes reconfiguration logic to dynamically change the initial configuration.
Enable IEEE 1588 Precision Time Protocol	**On/Off**	When you turn this option on, the core includes the soft FIFO for 1588, the associated logic, and enables the ports required for IEEE 1588 PTP.
Enable tx_pma_clkout port	**On/Off**	When you turn this option on, the `tx_pma_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_pma_clkout port	**On/Off**	When you turn this option on, the `rx_pma_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable tx_divclk port	**On/Off**	When you turn this option on, the `tx_divclk` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_divclk port	**On/Off**	When you turn this option on, the `rx_divclk` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable tx_clkout port	**On/Off**	When you turn this option on, the `tx_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_clkout port	**On/Off**	When you turn this option on, the `rx_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.

10GBASE-R Parameters

The 10GBASE-R parameters specify basic features of the 10GBASE-R PCS. The FEC options also allow you to specify the FEC ability.

Table 2-86: 10GBASE-R Parameters

Parameter Name	Options	Description
Reference clock frequency	**644.53125 MHz** **322.265625**	Specifies the input reference clock frequency. The default is **322.265625MHz**.
Enable additional control and status pins	**On/Off**	When you turn this option on, the core includes the `rx_block_lock` and `rx_hi_ber` ports.

Table 2-87: FEC Options

Parameter Name	Options	Description
Include FEC sublayer	**On/Off**	When you turn this option on, the core includes logic to implement FEC and a soft 10GBASE-R PCS.
Set `FEC_ability` bit on power up and reset	**On/Off**	When you turn this option on, the core sets the `Assert KR FEC Ability` bit (0xB0[16]) FEC ability bit during power up and reset, causing the core to advertise the FEC ability. This option is required for FEC functionality.
Set `FEC_Enable` bit on power up and reset	**On/Off**	When you turn this option **On**, the core sets the `KR FEC Request` bit (0xB0[18]) during power up and reset, causing the core to request the FEC ability during Auto Negotiation. This option is required for FEC functionality.

10M/100M/1Gb Ethernet Parameters

The 10M/100M/1GbE parameters allow you to specify options for the MII interface and the 1GbE data rate.

Table 2-88: 10M/100M/1Gb Ethernet

Parameter Name	Options	Description
Enable 1Gb Ethernet protocol	**On/Off**	When you turn this option on, the core includes the GMII interface and related logic.
Enable 10M/100Mb Ethernet functionality	**On/Off**	When you turn this option on, the core includes the MII PCS. It also supports 4-speed mode to implement a 10M/100M interface to the MAC for the GbE line rate.

Parameter Name	Options	Description
PHY ID (32 bits)	`32-bit value`	An optional 32-bit value that serves as a unique identifier for a particular type of PCS. The identifier includes the following components: • Bits 3-24 of the Organizationally Unique Identifier (OUI) assigned by the IEEE • 6-bit model number • 4-bit revision number If unused, do not change the default value which is 0x00000000.
PHY Core version (16 bits)	`16-bit value`	This is an optional 16-bit value identifies the PHY core version.

Speed Detection Parameters

By selecting the **Enable automatic speed detection** option in the MegaWizard Plug-In Manager, the PHY IP includes the sequencer module which implements the Parallel Detect function as described in the Ethernet specification.

Selecting the speed detection option gives the PHY the ability to detect to link partners that support 1G/10GbE but have disabled Auto-Negotiation. If you turn on the **Enable automatic speed detection** parameter, the PHY includes the sequencer block. During Auto-Negotiation, if AN cannot detect Differential Manchester Encoding (DME) pages from link partner, the Sequencer reconfigures to 1GE and 10GE modes (Speed/Parallel detection) until it detects a valid 1G or 10GbE pattern.

Table 2-89: Speed Detection

Parameter Name	Options	Description
Enable automatic speed detection	**On/Off**	When you turn this option **On**, the core includes the Sequencer block that sends reconfiguration requests to detect 1G or 10GbE when the Auto Negotiation block is not able detect AN data.
Avalon-MM clock frequency	`100-125 MHz`	Specifies the clock frequency for `phy_mgmt_clk`.
Link fail inhibit time for 10Gb Ethernet	`504 ms`	Specifies the time before `link_status` is set to FAIL or OK. A link fails if the `link_fail_inhibit_time` has expired before `link_status` is set to OK. The legal range is 500-510 ms. For more information, refer to *"Clause 73 Auto-Negotiation for Backplane Ethernet"* in *IEEE Std 802.3ap-2007*.
Link fail inhibit time for 1Gb Ethernet	`40-50 ms`	Specifies the time before `link_status` is set to FAIL or OK . A link fails if the link_fail_inhibit_time has expired before `link_status` is set to OK. The legal range is 40-50 ms. For more information, refer to *"Clause 73 Auto-Negotiation for Backplane Ethernet"* in *IEEE Std 802.3ap-2007*.

10GBASE-KR Auto-Negotiation Parameters

The 10GBASE-KR Auto-Negotiation parameters allow you to enable or disable auto negotiation.

Table 2-90: Auto Negotiation Settings

Name	Range	Description
Enable Auto Negotiation	**On/Off**	When you turn this option **On**, Auto Negotiation as defined in Clause 73 of the *IEEE Std 802.3ap-2007* is enabled.
Pause Ability-C0	**On/Off**	When you turn this option **On**, the core supports symmetric pauses as defined in *Annex 28B* of *Section 2 of IEEE Std 802.3-2008*.
Pause Ability-C1	**On/Off**	When you turn this option **On**, the core supports asymmetric pauses as defined in *Annex 28B* of *Section 2 of IEEE Std 802.3-2008*.

10GBASE-KR Link Training Parameters

The 10GBASE-KR variant provides parameters to customize the Link Training parameters.

Table 2-91: Link Training Parameters

In the following table, the exact correspondence between numerical values and voltages is pending characterization of the Native PHY.

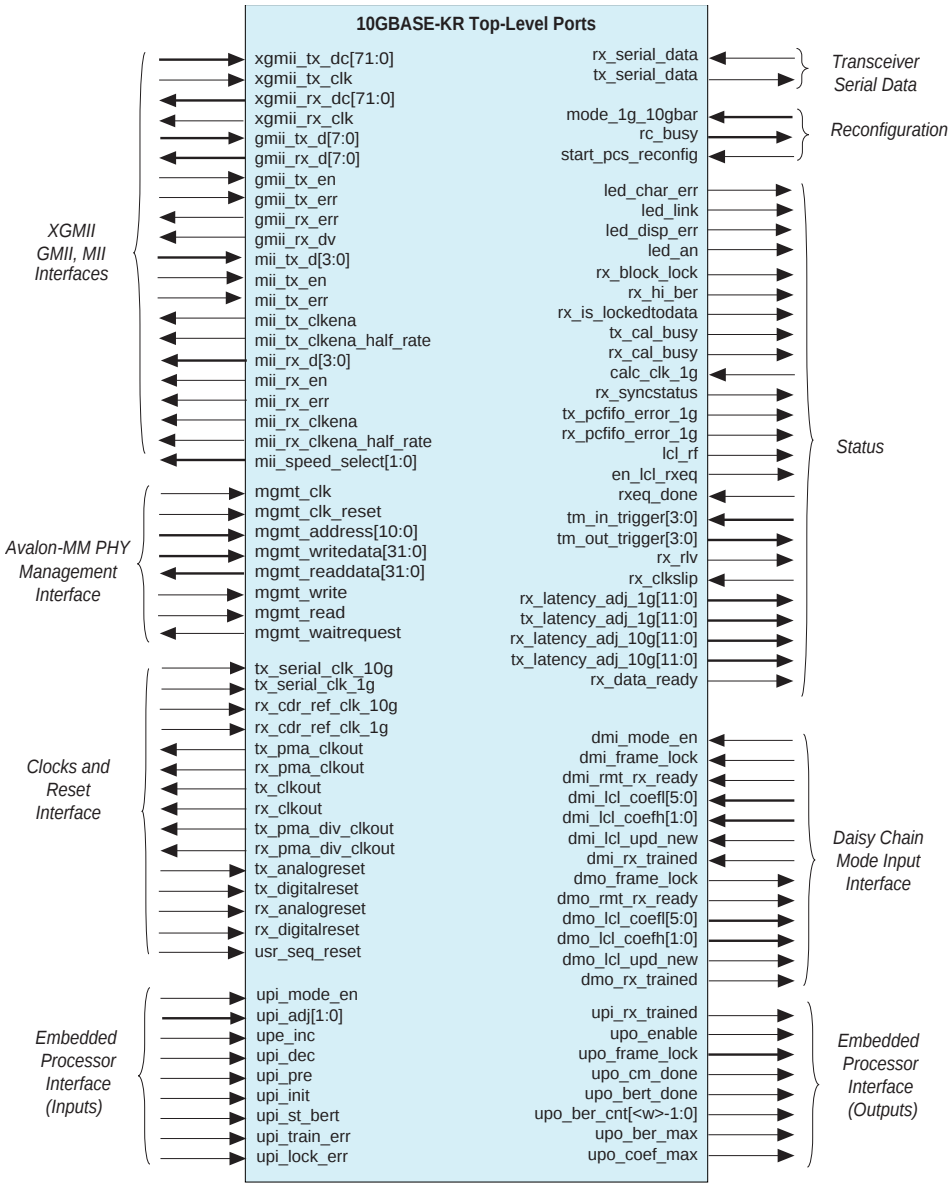
Name	Value	Description
Enable Link Training	**On/Off**	When you turn this option **On**, the core includes the link training module which configures the remote link-partner TX PMD for the lowest Bit Error Rate (BER). LT is defined in Clause 72 of *IEEE Std 802.3ap-2007*.
Enable daisy chain mode	**On/Off**	When you turn this option **On**, the core includes support for non-standard link configurations where the TX and RX interfaces connect to different link partners. This mode overrides the TX adaptation algorithm.
Enable microprocessor interface	**On/Off**	When you turn this option **On**, the core includes a microprocessor interface which enables the microprocessor mode for link training.
Maximum bit error count	**15, 31,63,127, 255,511,1023**	Specifies the maximum number of errors before the *Link Training Error* bit (0xD2, bit 4) is set indicating an unacceptable bit error rate. You can use this parameter to tune PMA settings. For example, if you see no difference in error rates between two different sets of PMA settings, you can increase the width of the bit error counter to determine if a larger counter enables you to distinguish between PMA settings.

Name	Value	Description
Number of frames to send before sending actual data	**127, 255**	Specifies the number of additional training frames the local link partner delivers to ensure that the link partner can correctly detect the local receiver state.
PMA Parameters		
VMAXRULE	0-63	Specifies the maximum V_{OD}. The default value is 60 which represents 1200 mV. The range is 0–1200 mV. If the backplane is short or lightly loaded, you can reduce the maximum value to limit the search space during link training.
VMINRULE	0-63	Specifies the minimum V_{OD}. The default value is 9 which represents 165 mV. If the backplane is long or heavily loaded, you can increase the minimum value to limit the search space during link training.
VODMINRULE	0-63	Specifies the minimum V_{OD} for the first tap. The default value is 24 which corresponds to 480 mV. This value needs to be equal or greater than the **VMINRULE** value and less than the **VMAXRULE** .
VPOSTRULE	0-31	Specifies the maximum value that the internal algorithm for pre-emphasis will ever test in determining the optimum post-tap setting. The default value is 31.
VPRERULE	0-15	Specifies the maximum value that the internal algorithm for pre-emphasis will ever test in determining the optimum pre-tap setting. The default value is 15.
PREMAINVAL	0-63	Specifies the Preset V_{OD} Value. Set by the Preset command as defined in Clause 72.6.10.2.3.1 of the link training protocol. This is the value from which the algorithm starts. The default value is 60.
PREPOSTVAL	0-31	Specifies the preset Pre-tap Value. The default value is 0.
PREPREVAL	0-15	Specifies the preset Post-tap value. The default value is 0.
INITMAINVAL	0-63	Specifies the Initial VOD Value. Set by the Initialize command in Clause 72.6.10.2.3.2 of the link training protocol. The default value is 52.
INITPOSTVAL	0-31	Specifies the initial first Post-tap value. The default value is 30.

Name	Value	Description
INITPREVAL	0-15	Specifies the Initial Pre-tap Value. The default value is 5.

10GBASE-KR PHY Interfaces

Figure 2-55: 10GBASE-KR Top-Level Signals



The block diagram shown in the GUI labels the external pins with the interface type and places the interface name inside the box. The interface type and name are used in the _hw.tcl file. If you turn on **Show signals**, the **block diagram** displays all top-level signal names. For more information about _hw.tcl files, refer to refer to the *Component Interface Tcl Reference* chapter in volume 1 of the *Quartus II Handbook*

Related Information

Component Interface Tcl Reference
For more information about _hw.tcl files.

Clock and Reset Interfaces

This topic defines the clock and reset signals.

The following table describes the clock and reset signals.

Table 2-92: Clock and Reset Signals

Signal Name	Direction	Description
`tx_serial_clk_10g`	Input	High speed clock from the 10G PLL to drive 10G PHY TX PMA. The frequency of this clock is 5.15625 GHz.
`tx_serial_clk_1g`	Input	High Speed clock from 1G PLL to drive the 1G PHY TX PMA. This clock is not required if GbE is not used. The frequency of this clock is 500 MHz.
`rx_cdr_ref_clk_10g`	Input	10G PHY RX PLL reference clock . This clock frequency can be 644.53125 MHz or 322.2656 MHz.
`rx_cdr_ref_clk_1g`	Input	1G PHY RX PLL reference clock. The frequency is 125 MHz. This clock is only required if 1G is enabled.
`tx_pma_clkout`	Output	This clock is used for the 1588 mode TX soft FIFO and 1G TX PCS parallel data. This clock frequency is 125 MHz for 1G and 257.81 MHz for 10G. This clock frequency is 161.13 MHz for 10G with FEC enabled.
`rx_pma_clkout`	Output	This clock is used for the 1588 mode RX soft FIFO and 1G RX PCS parallel data. This clock frequency is 125 MHz for 1G and 257.81 MHz for 10G. This clock frequency is 161.13 MHz for 10G with FEC enabled.
`tx_clkout`	Output	XGMII/GMII TX clock for the TX parallel data source interface. This clock frequency is 125 MHz in 1G mode and 257.81 MHz in 10G Mode, and 161.13 MHz with FEC enabled.
`rx_clkout`	Output	XGMII/GMII RX clock for the RX parallel data source interface. This clock frequency is 125 MHz in 1G mode and 257.81 in 10G Mode, and 161.13 MHz with FEC enabled.
`tx_pma_div_clkout`	Output	This is the divided 33 clock from the TX serializer. You can use this clock for the for `xgmii_tx_clk` or `xgmii_rx_clk`. This clock frequency is 125 MHz for 1G and 156.25 MHz for 10G. The frequencies are the same if you enable 1588 or FEC.
`rx_pma_div_clkout`	Output	This is the divided 33 clock from CDR recovered clock. This clock frequency is 125 MHz for 1G and 156.25 MHz for 10G. The frequencies are the same if you enable 1588 or FEC. This clock is not used for clock the 10G RX datapath. If PHY is reconfigured to 1G mode, this clock frequency changes to 125 MHz.

Signal Name	Direction	Description
tx_analogreset	Input	Resets the analog TX portion of the transceiver PHY.
tx_digitalreset	Input	Resets the digital TX portion of the transceiver PHY.
rx_analogreset	Input	Resets the analog RX portion of the transceiver PHY.
rx_digitalreset	Input	Resets the digital RX portion of the transceiver PHY.
usr_seq_reset	Input	Resets the sequencer. Initiates a PCS reconfiguration, and may restart AN, LT or both if these modes are enabled.

Related Information

- **Input Reference Clock Sources** on page 3-24
- **PLLs** on page 3-3

Data Interfaces

The following table describes the signals in the XGMII. GMII, and MII interfaces. The MAC drives the TX XGMII, GMII, and MII signals to the 1G/10GbE PHY. The 1G/10GbE PHY drives the RX XGMII, GMII, or MII signals to the MAC.

Table 2-93: SGMII and GMII Signals

Signal Name	Direction	Clock Domain	Description
1G/10GbE XGMII Data Interface			
xgmii_tx_dc[71:0]	Input	Synchronous to xgmii_tx_clk	XGMII data and control for 8 lanes. Each lane consists of 8 bits of data and 1 bit of control.
xgmii_tx_clk	Input	Clock signal	Clock for single data rate (SDR) XGMII TX interface to the MAC. It should connect to xgmii_rx_clk. This clock can be connected to the tx_div_clkout; however, Altera recommends that you connect it to a PLl for use with the Triple Speed Ethernet MegaCore Function. The frequency is 125 MHz for 1G and 156.25 MHz for 10G. This clock is driven from the MAC. The frequencies are the same if you enable 1588 or FEC.
xgmii_rx_dc[71:0]	Output	Synchronous to xgmii_rx_clk	RX XGMII data and control for 8 lanes. Each lane consists of 8 bits of data and 1 bit of control.

Signal Name	Direction	Clock Domain	Description
xgmii_rx_clk	Input	Clock signal	Clock for SDR XGMII RX interface to the MAC. This clock can be connected to the tx_div_clkout ; however, Altera recommends that you connect it to a PLl for use with the Triple Speed Ethernet MegaCore Function. The frequency is 125 MHz for 1G and 156.25 MHz for 10G. This clock is driven from the MAC. The frequencies are the same if you enable 1588 or FEC.
1G/10GbE GMII Data Interface			
gmii_tx_d[7:0]	Input	Synchronous to tx_pma_clkout	TX data for 1G mode. Synchronized to tx_pma_clkout clock. The TX PCS 8B/10B module encodes this data which is sent to link partner.
gmii_rx_d[7:0]	Output	Synchronous to rx_pma_clkout	RX data for 1G mode. Synchronized to rx_pma_clkout clock. The RX PCS 8B/10B decoders decodes this data and sends it to the MAC.
gmii_tx_en	Input	Synchronous to tx_pma_clkout	When asserted, indicates the start of a new frame. It should remain asserted until the last byte of data on the frame is present on gmii_tx_d.
gmii_tx_err	Input	Synchronous to tx_pma_clkout	When asserted, indicates an error. May be asserted at any time during a frame transfer to indicate an error in that frame.
gmii_rx_err	Output	Synchronous to rx_pma_clkout	When asserted, indicates an error. May be asserted at any time during a frame transfer to indicate an error in that frame.
gmii_rx_dv	Output	Synchronous to rx_pma_clkout	When asserted, indicates the start of a new frame. It remains asserted until the last byte of data on the frame is present on gmii_rx_d.
MII Data Interface			
mii_tx_d[3:0]	Input	Synchronous to tx_pma_clkout	TX data to be encoded and sent to the link partner. Synchronized to the tx_pma_clkout clock.
mii_tx_en	Input	Synchronous to tx_pma_clkout	When asserted, indicates the start of a new frame. mii_tx_enshould remain asserted until the last nibble of data on the frame is present on mii_tx_d[3:0].
mii_tx_err	Input	Synchronous to tx_pma_clkout	When asserted, it indicates an error in the frame. mii_tx_en should also be asserted for PHY to transmit invalid data.

Signal Name	Direction	Clock Domain	Description
mii_tx_clkena	Output	Clock signal	MII TX clock enable. This clock frequency has the following frequencies: • 25 MHz: For an effective rate of 100 Mbps • 2.5 MHz: For an effective rate of 10 Mbps
mii_tx_clkena_half_rate	Output	Clock signal	MII RX clock enable when the FPGA fabric runs at half the PCS frequency. This clock frequency has the following frequencies: • 12.5 MHz: For an effective rate of 100 Mbps • 1.25 MHz: For an effective rate of 10 Mbps
mii_rx_d[3:0]	Output	Synchronous to rx_pma_clkout	RX Data received from link partner. Synchronized to the rx_pma_clkout clock.
mii_rx_en	Output	Synchronous to rx_pma_clkout	When asserted, indicates that data onmii_rx_d[3:0] is valid.
mii_rx_err	Output	Synchronous to rx_pma_clkout	When asserted, it indicates an error in the frame.
mii_rx_clkena	Output	Clock signal	MII RX clock enable. This clock frequency has the following frequencies: • 25 MHz: For an effective rate of 100 Mbps • 2.5 MHz: For an effective rate of 10 Mbps
mii_rx_clkena_half_rate	Output	Clock signal	MII RX clock enable when the FPGA fabric runs at half the PCS frequency. This clock frequency has the following frequencies: • 12.5 MHz: For an effective rate of 100 Mbps • 1.25 MHz: For an effective rate of 10 Mbps
mii_speed_sel[1:0]	Output	Asynchronous signal	Specifies the SGMII (4-speed) mode. The following encodings are defined: • 2b'00: 10 Gbps • 2'b01: 1 Gbps • 2'b10: 100 Mbps • 2'b11: 10 Mbps

XGMII Mapping to Standard SDR XGMII Data

The 72-bit TX XGMII data bus format is different than the standard SDR XGMII interface. The following table shows the mapping of this non-standard format to the standard SDR XGMII interface

Table 2-94: TX XGMII Mapping to Standard SDR XGMII Interface

Signal Name	SDR XGMII Signal Name	Description
xgmii_tx_dc[7:0]	xgmii_sdr_data[7:0]	Lane 0 data
xgmii_tx_dc[8]	xgmii_sdr_ctrl[0]	Lane 0 control
xgmii_tx_dc[16:9]	xgmii_sdr_data[15:8]	Lane 1 data
xgmii_tx_dc[17]	xgmii_sdr_ctrl[1]	Lane 1 control
xgmii_tx_dc[25:18]	xgmii_sdr_data[23:16]	Lane 2 data
xgmii_tx_dc[26]	xgmii_sdr_ctrl[2]	Lane 2 control
xgmii_tx_dc[34:27]	xgmii_sdr_data[31:24]	Lane 3 data
xgmii_tx_dc[35]	xgmii_sdr_ctrl[3]	Lane 3 control
xgmii_tx_dc[43:36]	xgmii_sdr_data[39:32]	Lane 4 data
xgmii_tx_dc[44]	xgmii_sdr_ctrl[4]	Lane 4 control
xgmii_tx_dc[52:45]	xgmii_sdr_data[47:40]	Lane 5 data
xgmii_tx_dc[53]	xgmii_sdr_ctrl[5]	Lane 5 control
xgmii_tx_dc[61:54]	xgmii_sdr_data[55:48]	Lane 6 data
xgmii_tx_dc[62]	xgmii_sdr_ctrl[6]	Lane 6 control
xgmii_tx_dc[70:63]	xgmii_sdr_data[63:56]	Lane 7 data
xgmii_tx_dc[71]	xgmii_sdr_ctrl[7]	Lane 7 control

The 72-bit RX XGMII data bus format is different from the standard SDR XGMII interface. The following table shows the mapping of this non-standard format to the standard SDR XGMII interface:

Table 2-95: RX XGMII Mapping to Standard SDR XGMII Interface

Signal Name	XGMII Signal Name	Description
xgmii_rx_dc[7:0]	xgmii_sdr_data[7:0]	Lane 0 data
xgmii_rx_dc[8]	xgmii_sdr_ctrl[0]	Lane 0 control
xgmii_rx_dc[16:9]	xgmii_sdr_data[15:8]	Lane 1 data
xgmii_rx_dc[17]	xgmii_sdr_ctrl[1]	Lane 1 control
xgmii_rx_dc[25:18]	xgmii_sdr_data[23:16]	Lane 2 data
xgmii_rx_dc[26]	xgmii_sdr_ctrl[2]	Lane 2 control

Signal Name	XGMII Signal Name	Description
xgmii_rx_dc[34:27]	xgmii_sdr_data[31:24]	Lane 3 data
xgmii_rx_dc[35]	xgmii_sdr_ctrl[3]	Lane 3 control
xgmii_rx_dc[43:36]	xgmii_sdr_data[39:32]	Lane 4 data
xgmii_rx_dc[44]	xgmii_sdr_ctrl[4]	Lane 4 control
xgmii_rx_dc[52:45]	xgmii_sdr_data[47:40]	Lane 5 data
xgmii_rx_dc[53]	xgmii_sdr_ctrl[5]	Lane 5 control
xgmii_rx_dc[61:54]	xgmii_sdr_data[55:48]	Lane 6 data
xgmii_rx_dc[62]	xgmii_sdr_ctrl[6]	Lane 6 control
xgmii_rx_dc[70:63]	xgmii_sdr_data[63:56]	Lane 7 data
xgmii_rx_dc[71]	xgmii_sdr_ctrl[7]	Lane 7 control

Serial Data Interface

This topic describes the serial data interface.

Signal Name	Direction	Description
rx_serial_data	Input	RX serial input data
tx_serial_data	Output	TX serial output data

Control and Status Interfaces

The XGMII and GMII interface signals drive data to and from PHY.

Table 2-96: Control and Status Signals

Signal Name	Direction	Clock Domain	Description
led_char_err	Output	Synchronous to rx_clkout	10-bit character error. Asserted for one rx_clkout_1g cycle when an erroneous 10-bit character is detected.
led_link	Output	Synchronous to rx_clkout	When asserted, indicates successful link synchronization.
led_disp_err	Output	Synchronous to rx_clkout	Disparity error signal indicating a 10-bit running disparity error. Asserted for one rx_clkout_1g cycle when a disparity error is detected. A running disparity error indicates that more than the previous and perhaps the current received group had an error.
led_an	Output	Synchronous to rx_clkout	Clause 37 Auto-negotiation status. The PCS function asserts this signal when auto-negotiation completes.

Signal Name	Direction	Clock Domain	Description
rx_block_lock	Output	Synchronous to rx_clkout	Asserted to indicate that the block synchronizer has established synchronization.
rx_hi_ber	Output	Synchronous to rx_clkout	Asserted by the BER monitor block to indicate a Sync Header high bit error rate greater than 10^{-4}.
rx_is_lockedtodata	Output	Asynchronous signal	When asserted, indicates the RX channel is locked to input data.
tx_cal_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that the TX channel is being calibrated.
rx_cal_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that the RX channel is being calibrated.
calc_clk_1g	Input	Clock signal	This clock is used to calculate the latency of the soft 1G PCS block. This clock is only required for when you enable 1588 in 1G mode. Its frequency is 80 MHz. This clock should have the same PPM as the pll_ref_clk_1ginput.
rx_sync_status	Output	Synchronous to rx_clkout	When asserted, indicates the word aligner has aligned to in incoming word alignment pattern.
tx_pcfifo_error_1g	Output	Synchronous to tx_clkout	When asserted, indicates that the Standard PCS TX phase compensation FIFO is full.
rx_pcfifo_error_1g	Output	Synchronous to rx_clkout	When asserted, indicates that the Standard PCS RX phase compensation FIFO is full.
lcl_rf	Input	Synchronous to xgmii_tx_clk	When asserted, indicates a Remote Fault (RF).The MAC sends this fault signal to its link partner. Bit D13 of the Auto Negotiation Advanced Remote Fault register (0xC2) records this error.
rx_clkslip	Input	Asynchronous signal	When asserted, indicates that the deserializer has either skipped one serial bit or paused the serial clock for one cycle to achieve word alignment. As a result, the period of the parallel clock could be extended by 1 unit interval (UI) during the clock slip operation.
rx_latency_adj_1g[11:0]	Output	Synchronous to rx_clkout	When you enable 1588, this signal outputs the real time latency in GMII clock cycles (125 MHz) for the RX PCS and PMA datapath for 1G mode.
tx_latency_adj_1g[11:0]	Output	Synchronous to tx_clkout	When you enable 1588, this signal outputs real time latency in GMII clock cycles (125 MHz) for the TX PCS and PMA datapath for 1G mode.
rx_latency_adj_10g[11:0]	Output	Synchronous to rx_clkout	When you enable 1588, this signal outputs the real time latency in XGMII clock cycles (156.25 MHz) for the RX PCS and PMA datapath for 10G mode.
tx_latency_adj_10g[11:0]	Output	Synchronous to tx_clkout	When you enable 1588, this signal outputs real time latency in XGMII clock cycles (156.25 MHz) for the TX PCS and PMA datapath for 10G mode.

Signal Name	Direction	Clock Domain	Description
rx_data_ready	Output	Synchronous to rx_clkout	When asserted, indicates that the MAC can begin sending data to the PHY.

Daisy-Chain Interface Signals

The optional daisy-chain interface signals connect link partners using a daisy-chain topology.

Table 2-97: Daisy Chain Interface Signals

Signal Name	Direction	Description
dmi_mode_en	Input	When asserted, enable Daisy Chain mode.
dmi_frame_lock	Input	When asserted, the daisy chain state machine has locked to the training frames.
dmi_rmt_rx_ready	Input	Corresponds to bit 15 of Status report field. When asserted, the remote receiver.
dmi_lcl_coefl[5:0]	Input	Local update low bits[5:0]. In daisy-chained configurations, the local update coefficients substitute for the coefficients that would be set using Link Training.
dmi_lcl_coefh[1:0]	Input	Local update high bits[13:12]. In daisy-chained configurations, the local update coefficients substitute for the coefficients that would be set using Link Training.
dmi_lcl_upd_new	Input	When asserted, indicates a local update has occurred.
dmi_rx_trained	Input	When asserted, indicates that the state machine has finished local training.
dmo_frame_lock	Output	When asserted, indicates that the state machine has locked to the training frames.
dmo_rmt_rx_ready	Output	Corresponds to the link partner's remote receiver ready bit.
dmo_lcl_coefl[5:0]	Output	Local update low bits[5:0]. In daisy-chained configurations, the local update coefficients substitute for the coefficients that would be set using Link Training.
dmo_lcl_coefh[1:0]	Output	Local update high bits[13:12]. In daisy-chained configurations, the local update coefficients substitute for the coefficients that would be set using Link Training.
dmo_lcl_upd_new	Output	When asserted, indicates a local update has occurred.
dmo_rx_trained	Output	When asserted, indicates that the state machine has finished local training.

Embedded Processor Interface Signals

The optional embedded processor interface signals allow you to use the embedded processor mode of Link Training. This mode overrides the TX adaptation algorithm and allows an embedded processor to initialize the link.

Table 2-98: Embedded Processor Interface Signals

Signal Name	Direction	Description
upi_mode_en	Input	When asserted, enables embedded processor mode.
upi_adj[1:0]	Input	Selects the active tap. The following encodings are defined: • 2'b01: Main tap • 2'b10: Post-tap • 2'b11: Pre-tap
upi_inc	Input	When asserted, sends the increment command.
upi_dec	Input	When asserted, sends the decrement command.
upi_pre	Input	When asserted, sends the preset command.
upi_init	Input	When asserted, sends the initialize command.
upi_st_bert	Input	When asserted, starts the BER timer.
upi_train_err	Input	When asserted, indicates a training error.
upi_rx_trained	Input	When asserted, the local RX interface is trained.
upo_enable	Output	When asserted, indicates that the 10GBASE-KR PHY IP Core is ready to receive commands from the embedded processor.
upo_frame_lock	Output	When asserted, indicates the receiver has achieved training frame lock.
upo_cm_done	Output	When asserted, indicates the master state machine handshake is complete.
upo_bert_done	Output	When asserted, indicates the BER timer is at its maximum count.
upo_ber_cnt[<w>-1:0]	Output	Records the BER count.
upo_ber_max	Output	When asserted, the BER counter has rolled over.
upo_coef_max	Output	When asserted, indicates that the remote coefficients are at their maximum or minimum values.

Dynamic Reconfiguration Interface

You can use the dynamic reconfiguration interface signals to dynamically change between 1G and 10G data rates.

Table 2-99: Dynamic Reconfiguration Interface Signals

Signal Name	Direction	Clock Domain	Description
mode_1g_10gbar	Input	Synchronous to mgmt_clk	This signal indicates the requested mode for the channel. A 1 indicates 1G mode and a 0 indicates 10G mode.
rc_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that reconfiguration is in progress. Synchronous to the mgmt_clk. This signal is only exposed under the following conditions: • Turn off **Enable automatic speed detection** • Turn on **Enable internal PCS reconfiguration logic** • Turn on **Enable 1Gb Ethernet protocol**
start_pcs_reconfig	Input	Synchronous to mgmt_clk	When asserted, initiates reconfiguration of the PCS. Sampled with the mgmt_clk. This signal is only exposed under the following conditions: • Turn off **Enable automatic speed detection** • Turn on **Enable internal PCS reconfiguration logic** • Turn on **Enable 1Gb Ethernet protocol**

Avalon-MM Register Interface

The Avalon-MM slave interface signals provide access to all registers.

Table 2-100: Avalon-MM Interface Signals

Signal Name	Direction	Clock Domain	Description
mgmt_clk	Input	Clock	The clock signal that controls the Avalon-MM PHY management, interface. If you plan to use the same clock for the PHY management interface and transceiver reconfiguration, you must restrict the frequency range to 100-125 MHz to meet the specification for the transceiver reconfiguration clock.
mgmt_clk_reset	Input	Reset	Resets the PHY management interface. This signal is active high and level sensitive.
mgmt_addr[10:0]	Input	Synchronous to mgmt_clk	11-bit Avalon-MM address.
mgmt_writedata[31:0]	Input	Synchronous to mgmt_clk	Input data.
mgmt_readdata[31:0]	Output	Synchronous to mgmt_clk	Output data.
mgmt_write	Input	Synchronous to mgmt_clk	Write signal. Active high.
mgmt_read	Input	Synchronous to mgmt_clk	Read signal. Active high.

Signal Name	Direction	Clock Domain	Description
mgmt_waitrequest	Output	Synchronous to mgmt_clk	When asserted, indicates that the Avalon-MM slave interface is unable to respond to a read or write request. When asserted, control signals to the Avalon-MM slave interface must remain constant.

Related Information

Avalon Interface Specifications

10GBASE-KR PHY Register Definitions

The Avalon-MM slave interface signals provide access to the control and status registers.

The following table specifies the control and status registers that you can access over the Avalon-MM PHY management interface. A single address space provides access to all registers.

Note: Unless otherwise indicated, the default value of all registers is 0.

Note: Writing to reserved or undefined register addresses may have undefined side effects.

Table 2-101: 10GBASE-KR Register Definitions

Word Addr	Bit	R/W	Name	Description
0x4B0	0	RW	Reset SEQ	When set to 1, resets the 10GBASE-KR sequencer (auto rate detect logic), initiates a PCS reconfiguration, and may restart Auto-Negotiation, Link Training or both if AN and LT are enabled (10GBASE-KR mode). SEQ Force Mode[2:0] forces these modes. This reset self clears.
	1	RW	Disable AN Timer	Auto-Negotiation disable timer. If disabled (Disable AN Timer = 1) , AN may get stuck and require software support to remove the ABILITY_DETECT capability if the link partner does not include this feature. In addition, software may have to take the link out of loopback mode if the link is stuck in the ACKNOWLEDGE_DETECT state. To enable this timer set Disable AN Timer = 0.
	2	RW	Disable LF Timer	When set to 1, disables the Link Fault timer. When set to 0, the Link Fault timer is enabled.
	6:4	RW	SEQ Force Mode[2:0]	Forces the sequencer to a specific protocol. Must write the Reset SEQ bit to 1 for the Force to take effect. The following encodings are defined: • 3'b000: No force • 3'b001: GigE • 3'b010: Reserved • 3'b011: Reserved • 3'b100: 10GBASE-R • 3'b101: 10GBASE-KR • Others: Reserved
	16	RW	FEC ability	When set to 1, indicates that the FEC ability is supported. When the FEC ability changes, you must assert the Reset SEQ bit (0x4B0[0]) to renegotiate with the new value.
	18	RW	FEC request	When set to 1, indicates that FEC block is enabled. When this bit changes, you must assert the Reset SEQ bit (0x4B0[0]) to renegotiate with the new value.

Word Addr	Bit	R/W	Name	Description
0x4B1	0	R	SEQ Link Ready	When asserted, the sequencer is indicating that the link is ready.
	1	R	SEQ AN timeout	When asserted, the sequencer has had an Auto-Negotiation timeout. This bit is latched and is reset when the sequencer restarts Auto-Negotiation.
	2		SEQ LT timeout	When set, indicates that the Sequencer has had a timeout.
	13:8		SEQ Reconfig Mode[5:0]	Specifies the Sequencer mode for PCS reconfiguration. The following modes are defined: • Bit 8, mode[0]: AN mode • Bit 9, mode[1]: LT Mode • Bit 10, mode[2]: 10G data mode • Bit 11, mode[3]: Gige data mode • Bit 12, mode[4]: Reserved for XAUI • Bit13, mode[5]: 10G FEC mode
	16	R	KR FEC ability	When set to 1, indicates that the 10GBASE-KR PHY supports FEC. For more information, refer to *Clause 45.2.1.84 of IEEE 302.3ap-2007*.
	17	R	KR FEC err ind ability	When set to 1, indicates that the 10GBASE-KR PHY is capable of reporting FEC decoding errors to the PCS. For more information, refer to *Clause 74.8.3 of IEEE 302.3ap-2007*.
0x4B2	0:10	RW	Reserved	—
	11	RWSC	FEC TX Error Insert	Writing a 1 inserts 1 error pulse into the TX FEC depending on the Transcoder and Burst error settings.
	31:15	RWSC	Reserved	—
0x4C0	0	RW	AN enable	When set to 1, enables Auto-Negotiation function. The default value is 1. For additional information, refer to bit 7.0.12 in Clause 73.8 Management Register Requirements, of *IEEE 802.3ap-2007*.
	1	RW	AN base pages ctrl	When set to 1, the user base pages are enabled. You can send any arbitrary data via the user base page low/high bits. When set to 0, the user base pages are disabled and the state machine generates the base pages to send.
	2	RW	AN next pages ctrl	When set to 1, the user next pages are enabled. You can send any arbitrary data via the user next page low/high bits. When set to 0, the user next pages are disabled. The state machine generates the null message to send as next pages.
	3	R	Local device remote fault	When set to 1, the local device signals Remote Faults in the Auto-Negotiation pages. When set to 0 a fault has not occurred.

Word Addr	Bit	R/W	Name	Description
0x4C1	0	RW	Reset AN	When set to 1, resets all the 10GBASE-KR Auto-Negotiation state machines. This bit is self-clearing.
	4	RW	Restart AN TX SM	When set to 1, restarts the 10GBASE-KR TX state machine. This bit self clears. This bit is active only when the TX state machine is in the AN state. For more information, refer to bit 7.0.9 in Clause 73.8 Management Register Requirements of *IEEE 802.3ap-2007*.
	8	RW	AN Next Page	When asserted, new next page info is ready to send. The data is in the XNP TX registers. When 0, the TX interface sends null pages. This bit self clears. Next Page (NP) is encoded in bit D15 of Link Codeword. For more information, refer to Clause 73.6.9 and bit 7.16.15 of Clause 45.2.7.6 of *IEEE 802.3ap-2007*.

Word Addr	Bit	R/W	Name	Description
0x4C2	1	RO	AN page received	When set to 1, a page has been received. When 0, a page has not been received. The current value clears when the register is read. For more information, refer to bit 7.1.6 in Clause 73.8 of *IEEE 802.3ap-2007*.
	2	RO	AN Complete	When asserted, Auto-Negotiation has completed. When 0, Auto-Negotiation is in progress. For more information, refer to bit 7.1.5 in Clause 73.8 of *IEEE 802.3ap-2007*.
	3	RO	AN ADV Remote Fault	When set to 1, fault information has been sent to the link partner. When 0, a fault has not occurred. The current value clears when the register is read. Remote Fault (RF) is encoded in bit D13 of the base Link Codeword. For more information, refer to Clause 73.6.7 of and bit 7.16.13 of *IEEE 802.3ap-2007*.
	4	RO	AN RX SM Idle	When set to 1, the Auto-Negotiation state machine is in the idle state. Incoming data is not Clause 73 compatible. When 0, the Auto-Negotiation is in progress.
	5	RO	AN Ability	When set to 1, the transceiver PHY is able to perform Auto-Negotiation. When set to 0, the transceiver PHY i s not able to perform Auto-Negotiation. If your variant includes Auto-Negotiation, this bit is tied to 1. For more information, refer to bits 7.1.3 and 7.48.0 of Clause 45 of *IEEE 802.3ap-2007*.
	6	RO	AN Status	When set to 1, link is up. When 0, the link is down. The current value clears when the register is read. For more information, refer to bit 7.1.2 of Clause 45 of *IEEE 802.3ap-2007*.
	7	RO	LP AN Ability	When set to 1, the link partner is able to perform Auto-Negotiation. When 0, the link partner is not able to perform Auto-Negotiation. For more information, refer to bit 7.1.0 of Clause 45 of *IEEE 802.3ap-2007*.
	9	RO	Seq AN Failure	When set to 1, a sequencer Auto-Negotiation failure has been detected. When set to 0, a Auto-Negotiation failure has not been detected.
	17:12	RO	KR AN Link Ready[5:0]	Provides a one-hot encoding of an_receive_idle = true and link status for the supported link as described in Clause 73.10.1. The following encodings are defined: • 6'b000000: 1000BASE-KX • 6'b000001: Reserved • 6'b000100: 10GBASE-KR • 6'b001000: Reserved • 6'b010000: Reserved • 6'b100000: Reserved

Word Addr	Bit	R/W	Name	Description
0x4C3	15:0	RW	User base page low	The Auto-Negotiation TX state machine uses these bits if the AN base pages ctrl bit is set. The following bits are defined: • [15]: Next page bit • [14]: ACK which is controlled by the SM • [13]: Remote Fault bit • [12:10]: Pause bits • [9:5]: Echoed nonce which are set by the state machine • [4:0]: Selector Bit 49, the PRBS bit, is generated by the Auto-Negotiation TX state machine.
0x4C4	31:0	RW	User base page high	The Auto-Negotiation TX state machine uses these bits if the Auto-Negotiation base pages ctrl bit is set. The following bits are defined: • [29:5]: Correspond to page bit 45:21 which are the technology ability. • [4:0]: Correspond to bits 20:16 which are TX nonce bits. Bit 49, the PRBS bit, is generated by the Auto-Negotiation TX state machine.
0x4C5	15:0	RW	User Next page low	The Auto-Negotiation TX state machine uses these bits if the Auto-Negotiation next pages ctrl bit is set. The following bits are defined: • [15]: next page bit • [14]: ACK controlled by the state machine • [13]: Message Page (MP) bit • [12]: ACK2 bit • [11]: Toggle bit For more information, refer to Clause 73.7.7.1 Next Page encodings of *IEEE 802.3ap-2007*. Bit 49, the PRBS bit, is generated by the Auto-Negotiation TX state machine.
0x4C6	31:0	RW	User Next page high	The Auto-Negotiation TX state machine uses these bits if the Auto-Negotiation next pages ctrl bit is set. Bits [31:0] correspond to page bits [47:16]. Bit 49, the PRBS bit, is generated by the Auto-Negotiation TX state machine.
0x4C7	15:0	RO	LP base page low	The AN RX state machine received these bits from the link partner. The following bits are defined: • [15] Next page bit • [14] ACK which is controlled by the state machine • [13] RF bit • [12:10] Pause bits • [9:5] Echoed Nonce which are set by the state machine • [4:0] Selector

Word Addr	Bit	R/W	Name	Description
0x4C8	31:0	RO	LP base page high	The AN RX state machine received these bits from the link partner. The following bits are defined: • [31:30]: Reserved • [29:5]: Correspond to page bits [45:21] which are the technology ability • [4:0]: Correspond to bits [20:16] which are TX Nonce bits
0x4C9	15:0	RO	LP Next page low	The AN RX state machine receives these bits from the link partner. The following bits are defined: • [15]: Next page bit • [14]: ACK which is controlled by the state machine • [13]: MP bit • [12] ACK2 bit • [11] Toggle bit For more information, refer to Clause 73.7.7.1 Next Page encodings of IEEE 802.3ap-2007.
0x4CA	31:0	RO	LP Next page high	The AN RX state machine receives these bits from the link partner. Bits [31:0] correspond to page bits [47:16]
0x4CB	24:0	RO	AN LP ADV Tech_A[24:0]	Received technology ability field bits of Clause 73 Auto-Negotiation. The 10GBASE-KR PHY supports A0 and A2. The following protocols are defined: • A0 1000BASE-KX • A1 10GBASE-KX4 • A2 10GBASE-KR • A3 40GBASE-KR4 • A4 40GBASE-CR4 • A5 100GBASE-CR10 • A24:6 are reserved For more information, refer to Clause 73.6.4 and AN LP base page ability registers (7.19-7.21) of Clause 45 of *IEEE 802.3ap-2007*.
	27	RO	AN LP ADV Remote Fault	Received Remote Fault (RF) ability bits. RF is encoded in bit D13 of the base link codeword in Clause 73 AN. For more information, refer to Clause 73.6.7 and bits AN LP base page ability register AN LP base page ability registers (7.19-7.21) of Clause 45 of *IEEE 802.3ap-2007*.
	30:28	RO	AN LP ADV Pause Ability_C[2:0]	Received pause ability bits. Pause (C0:C1) is encoded in bits D11:D10 of the base link codeword in Clause 73 AN as follows: • C0 is the same as PAUSE as defined in Annex 28B • C1 is the same as ASM_DIR as defined in Annex 28B • C2 is reserved For more information, refer to bits AN LP base page ability registers (7.19-7.21) of Clause 45 of *IEEE 802.3ap-2007*.

Word Addr	Bit	R/W	Name	Description
0x4D0	0	RW	Link Training enable	When 1, enables the 10GBASE-KR start-up protocol. When 0, disables the 10GBASE-KR start-up protocol. The default value is 1. For more information, refer to Clause 72.6.10.3.1 and 10GBASE-KR PMD control register bit (1.150.1) of *IEEE 802.3ap-2007*.
	1	RW	dis_max_wait_ tmr	When set to 1, disables the LT max_wait_timer . Used for characterization mode when setting much longer BER timer values.
	2	RW	quick_mode	When set to 1, only the init and preset values are used to calculate the best BER.
	3	RW	pass_one	When set to 1, the BER algorithm considers more than the first local minimum when searching for the lowest BER. The default value is 1.
	7:4	RW	main_step_cnt [3:0]	Specifies the number of equalization steps for each main tap update. There are about 20 settings for the internal algorithm to test. The valid range is 1-15. The default value is 4'b0010.
	11:8	RW	prpo_step_cnt [3:0]	Specifies the number of equalization steps for each pre- and post- tap update. From 16-31 steps are possible. The default value is 4'b0001.
	13:12	RW	equal_cnt [1:0]	Adds hysteresis to the error count to avoid local minimums. The default value is 2'b01.
	16	RW	Ovride LP Coef enable	When set to 1, overrides the link partner's equalization coefficients; software changes the update commands sent to the link partner TX equalizer coefficients. When set to 0, uses the Link Training logic to determine the link partner coefficients. Used with 0x4D1 bit-4 and 0x4D4 bits[7:0].
	17	RW	Ovride Local RX Coef enable	When set to 1, overrides the local device equalization coefficients generation protocol. When set, the software changes the local TX equalizer coefficients. When set to 0, uses the update command received from the link partner to determine local device coefficients. Used with 0x4D1 bit-8 and 0x4D4 bits[23:16]. The default value is 1.

Word Addr	Bit	R/W	Name	Description
0x4D1	0	RW	Restart Link training	When set to 1, resets the 10GBASE-KR start-up protocol. When set to 0, continues normal operation. This bit self clears. For more information, refer to the state variable mr_restart_training as defined in Clause 72.6.10.3.1 and 10GBASE-KR PMD control register bit (1.150.0) *IEEE 802.3ap-2007*.
	4	RW	Updated TX Coef new	When sent to 1, there are new link partner coefficients available to send. The LT logic starts sending the new values set in 0x4D4 bits[7:0] to the remote device. When set to 0, continues normal operation. This bit self clears. Must enable this override in 0x4D0 bit16.
	8	RW	Updated RX coef new	When set to 1, new local device coefficients are available. The LT logic changes the local TX equalizer coefficients as specified in 0x4D4 bits[23:16]. When set to 0, continues normal operation. This bit self clears. Must enable the override in 0x4D0 bit17.
0x4D2	0	RO	Link Trained - Receiver status	When set to 1, the receiver is trained and is ready to receive data. When set to 0, receiver training is in progress. For more information, refer to the state variable rx_trained as defined in Clause 72.6.10.3.1 and bit 10GBASE-KR PMD control register bit 10GBASE_KR PMD status register bit (1.151.0) of *IEEE 802.3ap-2007*.
	1	RO	Link Training Frame lock	When set to 1, the training frame delineation has been detected. When set to 0, the training frame delineation has not been detected. For more information, refer to the state variable frame_lock as defined in Clause 72.6.10.3.1 and 10GBASE_KR PMD status register bit 10GBASE_KR PMD status register bit (1.151.1) of *IEEE 802.3ap-2007*.
	2	RO	Link Training Start-up protocol status	When set to 1, the start-up protocol is in progress. When set to 0, start-up protocol has completed. For more information, refer to the state training as defined in Clause 72.6.10.3.1 and 10GBASE_KR PMD status register bit (1.151.2) of *IEEE 802.3ap-2007*.
	3	RO	Link Training failure	When set to 1, a training failure has been detected. When set to 0, a training failure has not been detected For more information, refer to the state variable training_failure as defined in Clause 72.6.10.3.1 and bit 10GBASE_KR PMD status register bit (1.151.3) of IEEE 802.3ap-2007.
	4	RO	Link Training Error	When set to 1, excessive errors occurred during Link Training. When set to 0, the BER is acceptable.
	5	RO	Link Training Frame lock Error	When set to 1, indicates a frame lock was lost during Link Training. If the tap settings specified by the fields of 0x4D5 are the same as the initial parameter value, the frame lock error was unrecoverable.

Word Addr	Bit	R/W	Name	Description
0x4D3	9:0	RW	ber_time_frames	Specifies the number of training frames to examine for bit errors on the link for each step of the equalization settings. Used only when ber_time_k_frames is 0.The following values are defined: • A value of 2 is about 10^3 bytes • A value of 20 is about 10^4 bytes • A value of 200 is about 10^5 bytes The default value for simulation is 2'b11. The default value for hardware is 0.
	19:10	RW	ber_time_k_frames	Specifies the number of thousands of training frames to examine for bit errors on the link for each step of the equalization settings. Set *ber_time_m_frames = 0* for time/bits to match the following values: • A value of 3 is about 10^7 bits = about 1.3 ms • A value of 25 is about 10^8 bits = about 11ms • A value of 250 is about 10^9 bits = about 11 0ms The default value for simulation is 0. The default value for hardware is 0x415.
	29:20	RW	ber_time_m_frames	Specifies the number of millions of training frames to examine for bit errors on the link for each step of the equalization settings. Set *ber_time_k_frames = 4'd1000 = 0x43E8* for time/bits to match the following values: • A value of 3 is about 10^{10} bits = about 1.3 seconds • A value of 25 is about 10^{11} bits = about 11 seconds • A value of 250 is about 10^{12} bits = about 110 seconds

Word Addr	Bit	R/W	Name	Description

0x4D4	5:0	RO or RW	LD coefficient update[5:0]	Reflects the contents of the first 16-bit word of the training frame sent from the local device control channel. Normally, the bits in this register are read-only; however, when you override training by setting the Ovride Coef enable control bit, these bits become writeable. The following fields are defined:
				• [5: 4]: Coefficient (+1) update
				  • 2'b11: Reserved
				  • 2'b01: Increment
				  • 2'b10: Decrement
				  • 2'b00: Hold
				• [3:2]: Coefficient (0) update (same encoding as [5:4])
				• [1:0]: Coefficient (-1) update (same encoding as [5:4])
				For more information, refer to bit 10G BASE-KR LD coefficient update register bits (1.154.5:0) in Clause 45.2.1.80.3 of *IEEE 802.3ap-2007*.

	6	RO or RW	LD Initialize Coefficients	When set to 1, requests the link partner coefficients be set to configure the TX equalizer to its INITIALIZE state. When set to 0, continues normal operation. For more information, refer to 10G BASE-KR LD coefficient update register bits (1.154.12) in Clause 45.2.1.80.3 and Clause 72.6.10.2.3.2 of *IEEE 802.3ap-2007*.
	7	RO or RW	LD Preset Coefficients	When set to 1, requests the link partner coefficients be set to a state where equalization is turned off. When set to 0 the link operates normally. For more information, refer to bit 10G BASE-KR LD coefficient update register bit (1.154.13) in Clause 45.2.1.80.3 and Clause 72.6.10.2.3.2 of *IEEE 802.3ap-2007*.

	13:8	RO	LD coefficient status[5:0]	Status report register for the contents of the second, 16-bit word of the training frame most recently sent from the local device control channel. The following fields are defined:
				• [5:4]: Coefficient (post-tap)
				  • 2'b11: Maximum
				  • 2'b01: Minimum
				  • 2'b10: Updated
				  • 2'b00: Not updated
				• [3:2]: Coefficient (0) (same encoding as [5:4])
				• [1:0]: Coefficient (pre-tap) (same encoding as [5:4])
				For more information, refer to bit 10G BASE-KR LD status report register bit (1.155.5:0) in Clause 45.2.1.81 of *IEEE 802.3ap-2007*.

	14	RO	Link Training ready - LD Receiver ready	

Word Addr	Bit	R/W	Name	Description
				When set to 1, the local device receiver has determined that training is complete and is prepared to receive data. When set to 0, the local device receiver is requesting that training continue. Values for the receiver ready bit are defined in Clause 72.6.10.2.4.4. For more information refer to For more information, refer to bit 10G BASE-KR LD status report register bit (1.155.15) in Clause 45.2.1.81 of *IEEE 802.3ap-2007*.

	21:16	RO or RW	`LP coefficient update[5:0]`	Reflects the contents of the first 16-bit word of the training frame most recently received from the control channel.
				Normally the bits in this register are read only; however, when training is disabled by setting low the KR Training enable control bit, these bits become writeable. The following fields are defined:
				• [5: 4]: Coefficient (+1) update
				  • 2'b11: Reserved
				  • 2'b01: Increment
				  • 2'b10: Decrement
				  • 2'b00: Hold
				• [3:2]: Coefficient (0) update (same encoding as [5:4])
				• [1:0]: Coefficient (-1) update (same encoding as [5:4])
				For more information, refer to bit 10G BASE-KR LP coefficient update register bits (1.152.5:0) in Clause 45.2.1.78.3 of *IEEE 802.3ap-2007*.

	22	RO or RW	`LP Initialize Coefficients`	When set to 1, the local device transmit equalizer coefficients are set to the INITIALIZE state. When set to 0, normal operation continues. The function and values of the initialize bit are defined in Clause 72.6.10.2.3.2. For more information, refer to bit 10G BASE-KR LP coefficient update register bits (1.152.12) in Clause 45.2.1.78.3 of *IEEE 802.3ap-2007*.
	23	RO or RW	`LP Preset Coefficients`	When set to 1, The local device TX coefficients are set to a state where equalization is turned off. Preset coefficients are used. When set to 0, the local device operates normally. The function and values of the preset bit is defined in 72.6.10.2.3.1. The function and values of the initialize bit are defined in Clause 72.6.10.2.3.2. For more information, refer to bit 10G BASE-KR LP coefficient update register bits (1.152.13) in Clause 45.2.1.78.3 of IEEE 802.3ap-2007.
	29:24	RO	`LP coefficient status[5:0]`	

Word Addr	Bit	R/W	Name	Description

				Status report register reflects the contents of the second, 16-bit word of the training frame most recently received from the control channel: The following fields are defined:
				• [5:4]: Coefficient (+1)
				  • 2'b11: Maximum
				  • 2'b01: Minimum
				  • 2'b10: Updated
				  • 2'b00: Not updated
				• [3:2]: Coefficient (0) (same encoding as [5:4])
				• n [1:0]: Coefficient (-1) (same encoding as [5:4])
				For more information, refer to bit 10G BASE-KR LP status report register bits (1.153.5:0) in Clause 45.2.1.79 of *IEEE 802.3ap-2007*.

	30	RO	`LP Receiver ready`	When set to 1, the link partner receiver has determined that training is complete and is prepared to receive data. When set to 0, the link partner receiver is requesting that training continue. Values for the receiver ready bit are defined in Clause 72.6.10.2.4.4. For more information, refer to bit 10G BASE-KR LP status report register bits (1.153.15) in Clause 45.2.1.79 of *IEEE 802.3ap-2007*.
0x4D5	5:0	R	`LT` V_{OD} `setting`	Stores the most recent V_{OD} setting that LT specified using the Transceiver Reconfiguration Controller IP core. It reflects Link Partner commands to fine-tune the V_{OD}.
	12:8	R	`LT Post-tap setting`	Stores the most recent post-tap setting that LT specified using the Transceiver Reconfiguration Controller IP core. It reflects Link Partner commands to fine-tune the TX pre-emphasis taps.
	19:16	R	`LT Pre-tap setting`	Stores the most recent pre-tap setting that LT specified using the Transceiver Reconfiguration Controller IP core. It reflects Link Partner commands to fine-tune the TX pre-emphasis taps.

Word Addr	Bit	R/W	Name	Description
0x4D6	5:0	RW	LT VODMAX ovrd	Override value for the **VMAXRULE** parameter. When enabled, this value substitutes for the **VMAXRULE** to allow channel-by-channel override of the device settings. This only effects the local device TX output for the channel specified. This value must be greater than the **INITMAINVAL** parameter for proper operation. Note this will also override the **PREMAINVAL** parameter value.
	6	RW	LT VODMAX ovrd Enable	When set to 1, enables the override value for the **VMAXRULE** parameter stored in the LT VODMAX ovrd register field.
	13:8	RW	LT VODMin ovrd	Override value for the **VODMINRULE** parameter. When enabled, this value substitutes for the **VMINRULE** to allow channel-by-channel override of the device settings. This override only effects the local device TX output for this channel. The value to be substituted must be less than the **INITMAINVAL** parameter and greater than the **VMINRULE** parameter for proper operation.
	14	RW	LT VODMin ovrd Enable	When set to 1, enables the override value for the **VODMINRULE** parameter stored in the LT VODMin ovrd register field.
	20:16	RW	LT VPOST ovrd	Override value for the **VPOSTRULE** parameter. When enabled, this value substitutes for the **VPOSTRULE** to allow channel-by-channel override of the device settings. This override only effects the local device TX output for this channel. The value to be substituted must be greater than the **INITPOSTVAL** parameter for proper operation.
	21	RW	LT VPOST ovrd Enable	When set to 1, enables the override value for the **VPOSTRULE** parameter stored in the LT VPOST ovrd register field.
	27:24	RW	LT VPre ovrd	Override value for the **VPRERULE** parameter. When enabled, this value substitutes for the **VPOSTRULE** to allow channel-by-channel override of the device settings. This override only effects the local device TX output for this channel. The value greater than the **INITPREVAL** parameter for proper operation.
	28	RW	LT VPre ovrd Enable	When set to 1, enables the override value for the **VPRERULE** parameter stored in the LT VPre ovrd register field.

Word Addr	Bit	R/W	Name	Description
0x0DC 0x0DD 0x0DE 0x0DF	7:0	RSC	FEC Corrected Blocks	Counts the number of corrected FEC blocks. Resets to 0 when read. Otherwise, it holds at the maximum count and does not roll over. Refer to *Clause 74.8.4.1 of IEEE 802.3ap-2000* for details. The low-order byte of each register maps to the following bits of the 32 bit counter: • 0x0DC[7:0]: [7:0] • 0x0DD[7:0]: [15:8] • 0x0DE[7:0]: [23:16] • 0x0DF[7:0]:[31:24]
0x0E0 0x0E1 0x0E2 0x0E3	7:0	RSC	FEC Uncorrected Blocks	Counts the number of uncorrectable FEC blocks. Resets to 0 when read. Otherwise, it holds at the maximum count and does not roll over. Refer to *Clause 74.8.4.1 of IEEE 802.3ap-2000* for details. The low-order byte of each register maps to the following bits of the 32 bit counter: • 0x0E0[7:0]: [7:0] • 0x0E1[7:0]: [15:8] • 0x0E2[7:0]: [23:16] • 0x0E3[7:0]: [31:24]

Hard Transceiver PHY Registers

Table 2-102: Hard Transceiver PHY Registers

Addr	Bit	Access	Name	Description
0x000-0x3FF	[9:0]	RW	Access to HSSI registers	All registers in the PCS and PMA that you can dynamically reconfigure are in this address space. Refer to reconfiguration chapter for further information.

Enhanced PCS Registers

These registers provide Enhanced PCS status information.

Table 2-103: PCS Registers

Addr	Bit	Access	Name	Description
0x480	31:0	RW	Indirect_addr	Because the PHY implements a single channel, this register must remain at the default value of 0 to specify logical channel 0.
0x481	2	RW	RCLR_ERRBLK_CNT	Error Block Counter clear register. When set to 1, clears the RCLR_ERRBLK_CNT register. When set to 0, normal operation continues.
	3	RW	RCLR_BER_COUNT	BER Counter clear register. When set to 1, clears the RCLR_BER_COUNT register. When set to 0, normal operation continues.

Addr	Bit	Access	Name	Description
0x482	1	RO	HI_BER	High BER status. When set to 1, the PCS is reporting a high BER. When set to 0, the PCS is not reporting a high BER.
	2	RO	BLOCK_LOCK	Block lock status. When set to 1, the PCS is locked to received blocks. When set to 0, the PCS is not locked to received blocks.
	3	RO	TX_FULL	When set to 1, the TX_FIFO is full.
	4	RO	RX_FULL	When set to 1, the RX_FIFO is full.
	5	RO	RX_SYNC_HEAD_ERROR	When set to 1, indicates an RX synchronization error.
	6	RO	RX_SCRAMBLER_ERROR	When set to 1, indicates an RX scrambler error.
	7	RO	Rx_DATA_READY	When set to 1, indicates the PHY is ready to receive data.

PMA Registers

The PMA registers allow you to reset the PMA, customize the TX and RX serial data interface, and provide status information.

Address	Bit	R/W	Name	Description
0x422	[<p>-1:0]	RO	pma_tx_pll_is_locked	Indicates that the TX PLL is locked to the input reference clock. <p> is the number of PLLs.
0x444	1	RW	reset_tx_digital	Writing a 1 causes the internal TX digital reset signal to be asserted. You must write a 0 to clear the reset condition.
	2	RW	reset_rx_analog	Writing a 1 causes the internal RX analog reset signal to be asserted. You must write a 0 to clear the reset condition.
	3	RW	reset_rx_digital	Writing a 1 causes the internal RX digital reset signal to be asserted. You must write a 0 to clear the reset condition.
0x461	[31:0]	RW	phy_serial_loopback	Writing a 1 puts the channel in serial loopback mode.
0x464	[31:0]	RW	pma_rx_set_locktodata	When set, programs the RX CDR PLL to lock to the incoming data.
0x465	[31:0]	RW	pma_rx_set_locktoref	When set, programs the RX clock data recovery (CDR) PLL to lock to the reference clock.
0x466	[31:0]	RO	pma_rx_is_lockedtodata	When asserted, indicates that the RX CDR PLL is locked to the RX data, and that the RX CDR has changed from LTR to LTD mode.
0x467	[31:0]	RO	pma_rx_is_lockedtoref	When asserted, indicates that the RX CDR PLL is locked to the reference clock.

Address	Bit	R/W	Name	Description
0x4A8	0	RW	tx_invpolarity	When set to 1, the TX interface inverts the polarity of the TX data. Inverted TX data is input to the 8B/10B encoder.
	1	RW	rx_invpolarity	When set to 1, the RX channels inverts the polarity of the received data. Inverted RX data is input to the 8B/10B decoder.
	2	RW	rx_bitreversal_enable	When set to 1, enables bit reversal on the RX interface. The RX data is input to the word aligner.
	3	RW	rx_bytereversal_enable	When set, enables byte reversal on the RX interface. The RX data is input to the byte deserializer.
	4	RW	force_electrical_idle	When set to 1, forces the TX outputs to electrical idle.
0x4A9	0	R	rx_syncstatus	When set to 1, indicates that the word aligner is synchronized to incoming data.
	1	R	rx_patterndetect	When set to 1, indicates the 1G word aligner has detected a comma.
	2	R	rx_rlv	When set to 1, indicates a run length violation.
	3	R	rx_rmfifodatainserted	When set to 1, indicates the rate match FIFO inserted code group.
	4	R	rx_rmfifodatadeleted	When set to 1, indicates that rate match FIFO deleted code group.
	5	R	rx_disperr	When set to 1, indicates an RX 8B/10B disparity error.
	6	R	rx_errdetect	When set to 1, indicates an RX 8B/10B error detected.

Arria 10 GMII PCS Registers

This topic describes the GMII PCS registers.

Addr	Bit	R/W	Name	Description
0x490	9	RW	RESTART_AUTO_NEGOTIATION	Set this bit to 1 to restart the Clause 37 Auto-Negotiation sequence. For normal operation, set this bit to 0 which is the default value. This bit is self-clearing.
	12	RW	AUTO_NEGOTIATION_ENABLE	Set this bit to 1 to enable Clause 37 Auto-Negotiation. The default value is 1.
	15	RW	Reset	Set this bit to 1 to generate a synchronous reset pulse which resets all the PCS state machines, comma detection function, and the 8B/10B encoder and decoder. For normal operation, set this bit to 0. This bit self clears.

Addr	Bit	R/W	Name	Description
0x491	2	R	LINK_STATUS	A value of 1 indicates that a valid link is operating. A value of 0 indicates an invalid link. If link synchronization is lost, this bit is 0.
	3	R	AUTO_NEGOTIATION_ABILITY	A value of 1 indicates that the PCS function supports Clause 37 Auto-Negotiation.
	5	R	AUTO_NEGOTIATION_COMPLETE	A value of 1 indicates the following status: • The Auto-Negotiation process is complete. • The Auto-Negotiation control registers are valid.
0x494	5	RW	FD	Full-duplex mode enable for the local device. Set to 1 for full-duplex support.
	6	RW	HD	Half-duplex mode enable for the local device. Set to 1 for half-duplex support. This bit should always be set to 0.
	8:7	RW	PS2,PS1	Pause support for local device. The following encodings are defined for PS1/PS2: • 2'b00: Pause is not supported • 2'b0 1: Asymmetric pause toward link partner • 2'b10: Symmetric pause • 2'b11: Pause is supported on TX and RX
	13:12	RW	RF2,RF1	Remote fault condition for local device. The following encodings are defined for RF1/RF2: • 2'b00: No error, link is valid (reset condition) • 2'b0 1: Offline • 2'b10: Failure condition • 2'b11: Auto-negotiation error
	14	R0	ACK	Acknowledge for local device. A value of 1 indicates that the device has received three consecutive matching ability values from its link partner.
	15	RW	NP	Next page. In the device ability register, this bit is always set to 0.

Addr	Bit	R/W	Name	Description
0x495	5	R	FD	Full-duplex mode enable for the link partner. This bit should always be 1 because only full duplex is supported.
	6	R	HD	Half-duplex mode enable for the link partner. A value of 1 indicates support for half duplex. This bit should always be 0 because half-duplex mode is not supported.
	8:7	R	PS2,PS1	Specifies pause support for link partner. The following encodings are defined for PS1/PS2: • 2'b00: Pause is not supported • 2'b0 1: Asymmetric pause toward link partner • 2'b10: Symmetric pause • 2'b11: Pause is supported on TX and RX
	13:12	R	RF2,RF1	Remote fault condition for link partner. The following encodings are defined for RF1/RF2: • 2'b00: No error, link is valid (reset condition) • 2'b0 1: Offline • 2'b10: Failure condition • 2'b11: Auto-negotiation error
	14	R	ACK	Acknowledge for link partner. A value of 1 indicates that the device has received three consecutive matching ability values from its link partner.
	15	R	NP	Next page. In link partner register. When set to 0, the link partner has a Next Page to send. When set to 1, the link partner does not a Next Page. Next Page is not supported in Auto Negotiation.
0x496	0	R	LINK_ PARTNER_ AUTO_ NEGOTIATION_ ABLE	Set set to 1, indicates that the link partner supports auto negotiation. The default value is 0.
	1	R	PAGE_RECEIVE	A value of 1 indicates that a new page has been received with new partner ability available in the register partner ability. The default value is 0 when the system management agent performs a read access.
0x4A2	15:0	RW	AN link timer[15:0]	Low-order 16 bits of the 21-bit auto-negotiation link timer. Each timer step corresponds to 8ns (assuming a 125 MHz clock). The total timer corresponds to 16 ms. The reset value sets the timer to 10 ms for hardware mode and 10 us for simulation mode.
0x4A3	4:0	RW	AN link timer[4:0]	High-order 5 bits of the 21-bit auto-negotiation link timer.

Addr	Bit	R/W	Name	Description
0x4A4	0	RW	SGMII_ENA	Determines the PCS function operating mode. Setting this bit to 1b'1 enables SGMII mode. Setting this bit to 1b'0 enables 1000BASE-X gigabit mode.
	1	RW	USE_SGMII_AN	In SGMII mode, setting this bit to 1b'1 causes the PCS to be configured with the link partner abilities advertised during auto-negotiation. If this bit is set to 1b'0, the PCS function should be configured with the SGMII_SPEED and SGMII_DUPLEX bits.
	3:2	RW	SGMII_SPEED	SGMII speed. When the PCS operates in SGMII mode (SGMII_ENA = 1) and is not programmed for automatic configuration (USE_SGMII_AN = 0), the following encodings specify the speed : • 2'b00: 10 Mbps • 2'b01: 100 Mbps • 2'b10: Gigabit • 2'b11: Reserved These bits are not used when SGMII_ENA = 0or USE_SGMII_AN = 1.

Creating a 10GBASE-KR Design

Here are the steps you must take to create a 10GBASE-KR design using the Backplane Ethernet PHY IP.

1. Generate the 10GBASE-KR PHY with the required parameterization.

   Unlike in the 10GBASE-KR PHY IP Core for Stratix V devices, for Arria 10 devices, the Reconfiguration Block is included 10GBASE-KR PHY. This Reconfiguration Block provides the Avalon-MM master that reads and writes to PHY registers.
2. Instantiate a reset controller using the Transceiver Reset Controller Megafunction in the MegaWizard Plug-In Manager. Connect the power and reset signals between the 10GBASE-KR PHY and the reset controller.
3. Instantiate one TX PLL for the 1G data rate and one TX PLL for the 10G data rate. Connect the high speed serial clock and PLL lock signals between 10GBASE-KR PHY and TX PLLs. For the 1G data rate you can use either fPLL, or ATX, or CMU PLL. For the 10G data rate you can use ATX PLL or CMU PLL.
4. Generate a fractional PLL to create the 156.25 MHz XGMII clock from the 10G reference clock.
5. Use the tx_pma_divclk from the 10GBASE-KR PHY or generate a fPLL to create the 156.25 MHz XGMII clock from the 10G reference clock.

   Unlike in the 10GBASE-KR PHY IP core for Stratix V devices, no Memory Initialization Files (**.mif**) are required for the 10GBASE-KR design in Arria 10 devices.
6. Complete the design by creating a top level module to connect all the IP (10GBASE-KR PHY IP, PLL IP and Reset Controller) blocks.

Related Information

- **fPLL** on page 3-11
- **CMU PLL** on page 3-19

- **ATX PLL** on page 3-3
- **Using the Altera Transceiver PHY Reset Controller** on page 4-8
- **10GBASE-KR Functional Description** on page 2-106

Design Guidelines

Consider the following guidelines while designing with the 10GBASE-KR PHY IP.

Using the 10GBASE-KR PHY without the Sequencer

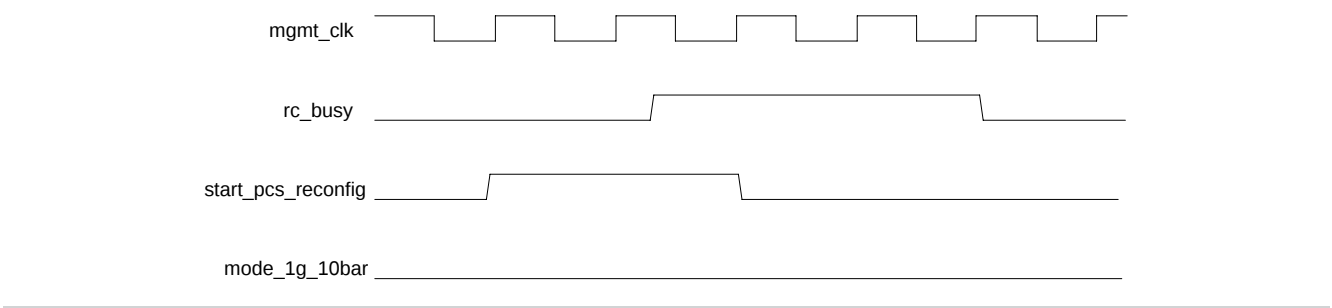
The sequencer brings up channel-based initial datapath and performs parallel detection. To use 10GBASE-KR PHY without the Sequencer, turn off the **Enable automatic speed detection** parameter.

Turning off the sequencer results in the following additional ports:

- `rc_busy`
- `start_pcs_reconfig`
- `mode_1g_10gbar`

These ports are used to request manual reconfiguration. The following figure shows the relationship of these ports for requesting 1G and 10G configuration. The reconfiguration is complete when the `rc_busy` signal goes low.

Figure 2-56: Timing for Reconfiguration without the Sequencer



1588 Delay Requirements

The 1588 protocol requires symmetric delays or known asymmetric delays for all external connections. In calculating the delays for all external connections, you must consider the delay contributions of the

following elements:

- The PCB traces
- The backplane traces
- The delay through connectors
- The delay through cables

Accurate calculation of the channel-to-channel delay is important in ensuring the overall system accuracy.

Channel Placement Guidelines

The channels of multi-channel 1G/10G designs do not need to be placed contiguously. However, channels instantiated in different transceiver banks require PLLs in the same bank. Currently the Quartus II software version 13.1 for Arria 10 devices does not support PMA bonding for 10GBASE-KR PHY IP.

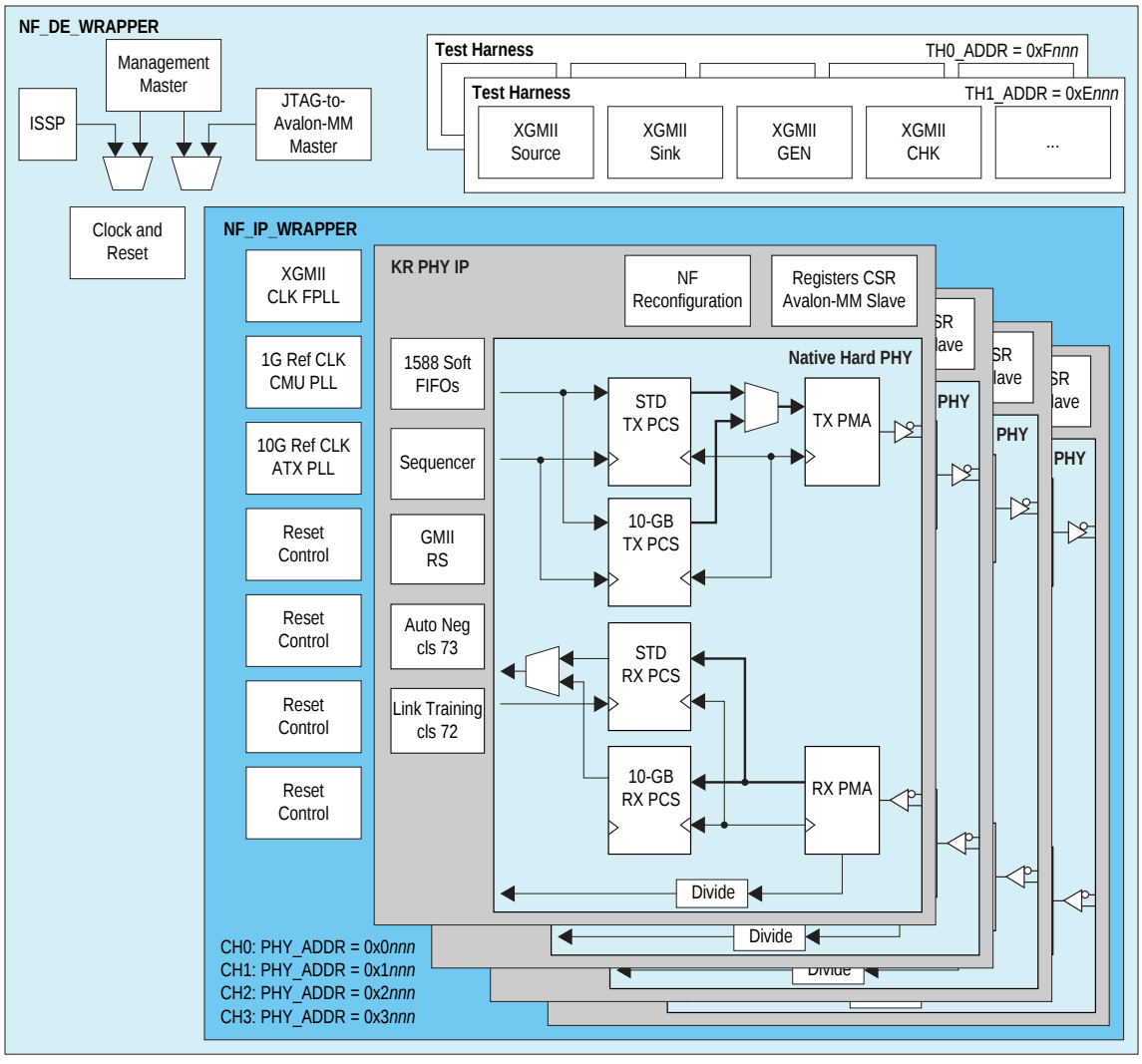
Design Example

Altera provides a MAC and PHY design example and a PHY-only design example to assist you in integrating your Ethernet PHY IP into your complete design.

The MAC and PHY design example instantiates the 1G/10GbE PHY IP along with the 1G/10G Ethernet MAC and supporting logic. It is part of the Quartus II installation and is located in the *<quartus2_install_dir>*/**ip** subdirectory. For more information about this example design, refer to the *Ethernet MAC MegaCore Function User Guide*.

The PHY-only design example instantiates the 1G/10GbE and 10GBASE-KR PHY IP and its supporting logic. This design example is available on the Altera Wiki. The following figure shows the block diagram of the 1G/10GbE PHY-only design example. The design example default configuration includes two channels for Backplane Ethernet and two channels for line-side (1G/10G) applications.

Figure 2-57: PHY-Only Design Example with Two Backplane Ethernet and Two Line-Side (1G/10G) Ethernet Channels



Related Information

- **Arria 10 Transceiver PHY Design Examples**
- **Ethernet MAC MegaCore Function User Guide**
  For more information about this design example.

Simulation Support

The 1G/10GbE and 10GBASE-KR PHY IP Core supports ModelSim Verilog and ModelSim VHDL, VCS Verilog, and VCS VHDL simulation. Arria 10 devices also support NCSIM Verilog and NCSIM VHDL simulation. The MegaWizard Plug-In Manager generates an IP functional simulation model when you press the **Finish** button.

TimeQuest Timing Constraints

To pass timing analysis, you must decouple the clocks in different time domains. The necessary Synopsys Design Constraints File (**.sdc**) timing constraints for the are included in the top-level wrapper file.

1G/10 Gbps Ethernet PHY IP Core

Ethernet standard comprises many different PHY standards with variations in signal transmission medium and data rates. The 1G/10Gbps Ethernet PHY IP core targets 10MB/100MB/1G/10GbE data rates with one core. This Ethernet PHY interfaces to 1G/10GbE dual speed SFP+ pluggable modules, 10MB–10GbE 10GBASE-T, and 10MB/100MB/1000MB 1000BASE-T copper external PHY devices to drive CAT-6/7 shielded twisted pair cables, and chip-to-chip interfaces.

The 1G/10 Gbps Ethernet PHY MegaCore® (1G/10GbE) Function allows you to support the following features of Ethernet standards:

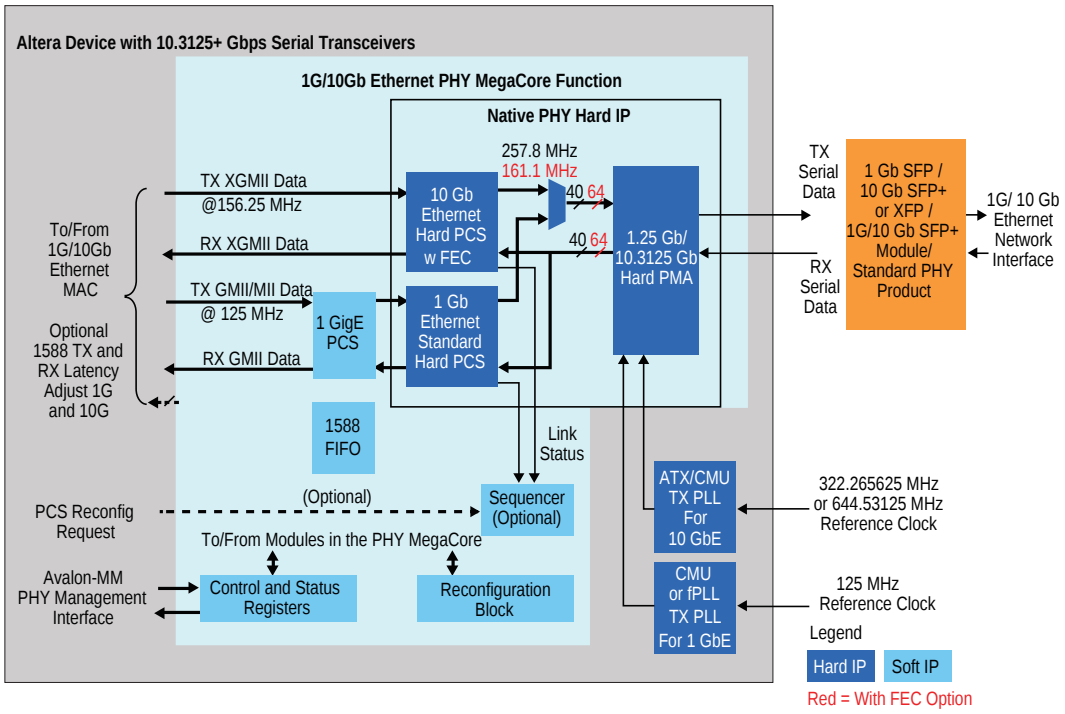
- 1 GbE protocol as defined in *Clause 36* of the *IEEE 802.3 2008 Standard*.
- GMII to connect PHY with media access control (MAC) as defined in *Clause 35* of the *IEEE 802.3 2008 Standard*
- Gigabit Ethernet Auto-negotiation as defined in*Clause 37* of the *IEEE 802.3 2008 Standard*
- 10GBASE-R Ethernet protocol as defined in*Clause 49* of the *IEEE 802.3 2008 Standard*
- Single data rate (64 data bits and 8 control bits) XGMII to provide simple and inexpensive interconnection between the MAC and the PHY as defined in *Clause 46* of the *IEEE 802.3 2008 Standard*
- SGMII 10MB/100MB/1000MB serial interface
- Forward Error correction(FEC) as defined in *Clause 74* of the *IEEE 802.3 2008 Standard*
- Precision time protocol (PTP) as defined in the *IEEE 1588 Standard*
- 10M/100Mbps MII to connect physical media with the MAC as defined in *Clause 22* of the *IEEE 802.1 2008 Standard*

The 1G/10Gbps Ethernet PHY MegaCore Function allows you to implement the 1GbE protocol using the Standard PCS and 10GbE protocol using Enhanced PCS and PMA. You can switch dynamically between the 1G and 10G data rates using dynamic reconfiguration to reprogram the core. Or, you can use the speed detection option to automatically switch data rates based on received data. The 1G/10Gbps Ethernet PHY also supports a SGMII (10MB/100MB/1000MB) interface.

The following figure shows the top-level modules of the 1G/10GbE PHY IP. As this figure indicates, the 1G/10 Gbps Ethernet PHY connects to a separately instantiated MAC. The Enhanced PCS receives and

transmits XGMII data. The Standard PCS receives and transmits GMII data. An Avalon® Memory-Mapped (Avalon-MM) slave interface provides access to PCS registers. The PMA receives and transmits serial data.

Figure 2-58: Level Modules of the 1G/10GbE PHY MegaCore Function


Altera Device with 10.3125+ Gbps Serial Transceivers
1G/10Gb Ethernet PHY MegaCore Function
Native PHY Hard IP
257.8 MHz
161.1 MHz
TX XGMII Data
@156.25 MHz
RX XGMII Data
TX GMII/MII Data
@ 125 MHz
RX GMII Data
To/From 1G/10Gb Ethernet MAC
Optional 1588 TX and RX Latency Adjust 1G and 10G
10 Gb Ethernet Hard PCS w FEC
1 GigE PCS
1 Gb Ethernet Standard Hard PCS
1588 FIFO
40 64
40 64
1.25 Gb/ 10.3125 Gb Hard PMA
Link Status
(Optional)
Sequencer (Optional)
PCS Reconfig Request
To/From Modules in the PHY MegaCore
Avalon-MM PHY Management Interface
Control and Status Registers
Reconfiguration Block
ATX/CMU TX PLL For 10 GbE
CMU or fPLL TX PLL For 1 GbE
322.265625 MHz or 644.53125 MHz Reference Clock
125 MHz Reference Clock
TX Serial Data
RX Serial Data
1 Gb SFP / 10 Gb SFP+ or XFP / 1G/10 Gb SFP+ Module/ Standard PHY Product
1G/ 10 Gb Ethernet Network Interface
Legend
Hard IP
Soft IP
Red = With FEC Option

An Avalon-MM slave interface provides access to the 1G/10GbE PHY IP Core registers. These registers control many of the functions of the other blocks. Many of these bits are defined in *Clause 45 of IEEE Std 802.3ap-2008*.

Related Information

- **IEEE Std 802.3ap-2008 Standard**
- **Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems**

1G/10GbE PHY Release Information

This topic provides information about this release of the 1G/10GbE PHY IP Core.

Table 2-104: 1G/10GbE Release Information

Item	Description
Version	13.1
Release Date	November 2013
Ordering Codes	IP-1G10GBASER PHY (primary)
Product ID	0106

Item	Description
Vendor ID	6AF7

1G/10GbE PHY Performance and Resource Utilization

This topic provides performance and resource utilization for the 1G/10GbE PHY IP core in Arria 10 devices.

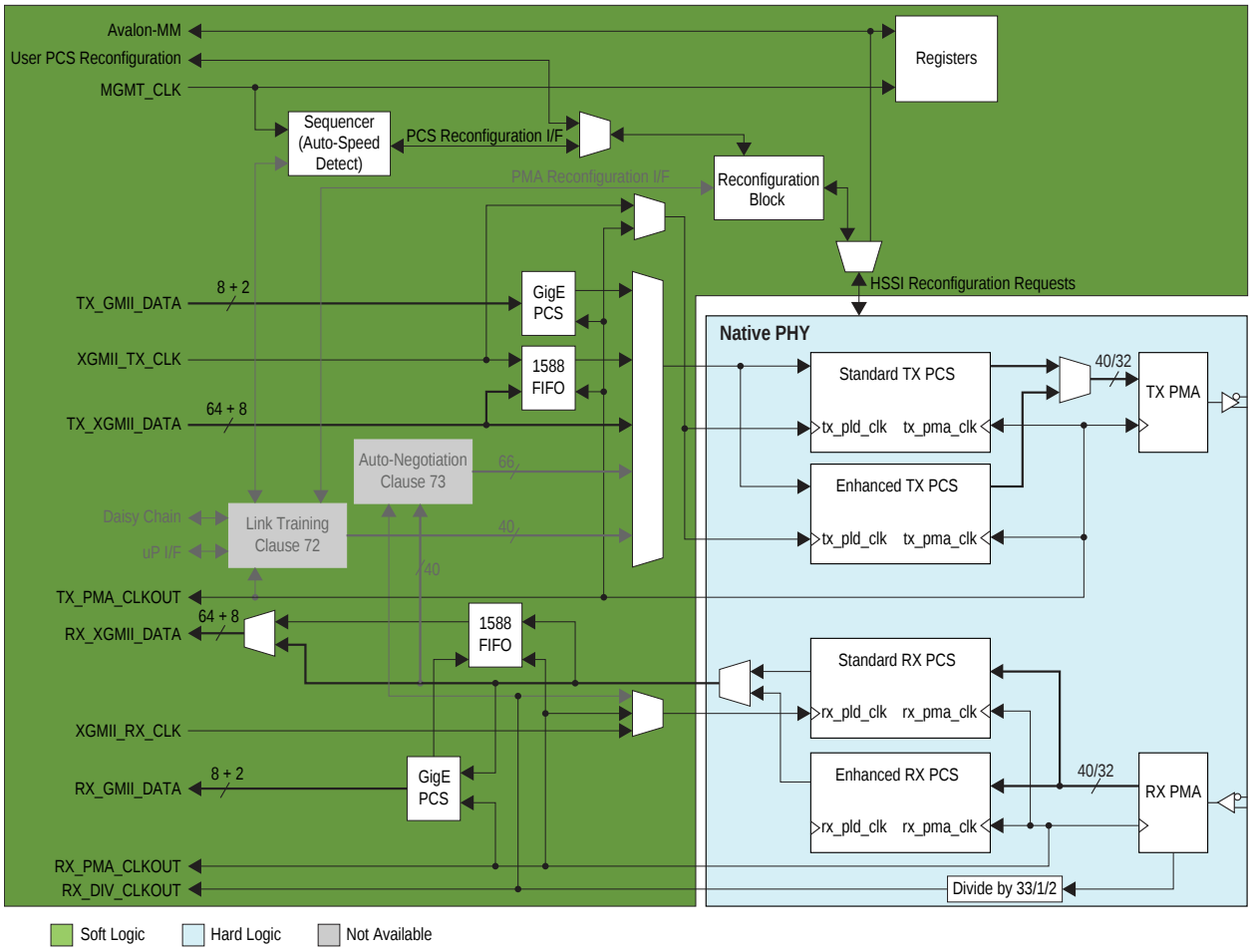
The following table shows the typical expected resource utilization for selected configurations using the Quartus II software Arria 10 Edition v13.1. The numbers of ALMs and logic registers are rounded up to the nearest 100.

Table 2-105: 1G/10GbE PHY Performance and Resource Utilization

Variant	ALMs	ALUTs	Registers	M20K
1G/10GbE PHY with 1588	2950	3900	4950	5
1G/10GbE PHY	750	1100	1300	2
1G/10GbE PHY with FEC	750	1100	1300	2

1G/10GbE PHY Functional Description

Figure 2-59: 10G/10GbE PHY Block Diagram



As this figure illustrates, the 1G/10GbE Ethernet PHY IP Core is built using the Native PHY. It includes the following modules:

Standard and Enhanced PCS Datapaths

The Standard PCS supports the 1G protocol. The Enhanced PCS supports 10GBASE-R. Refer to the Standard PCS and Enhanced PCS architecture chapters for more details on how these blocks support 1G,10G protocols and FEC.

Sequencer

The Sequencer controls the start-up sequence of the PHY IP, including reset and power-on. It selects which PCS (1G or 10G) and PMA interface is active. The Sequencer interfaces to the reconfiguration block to request a reconfigurations to change from one data rate to the other data rate.

GigE PCS

The GigE PCS includes the GMII interface and *Clause 37* auto negotiation and SGMII functionality.

1588 FIFO

The 1588 FIFO has an XGMII-like interface for both input and outputs. The 1588 FIFO includes the latency adjust information that the 1588 logic in the MAC requires.

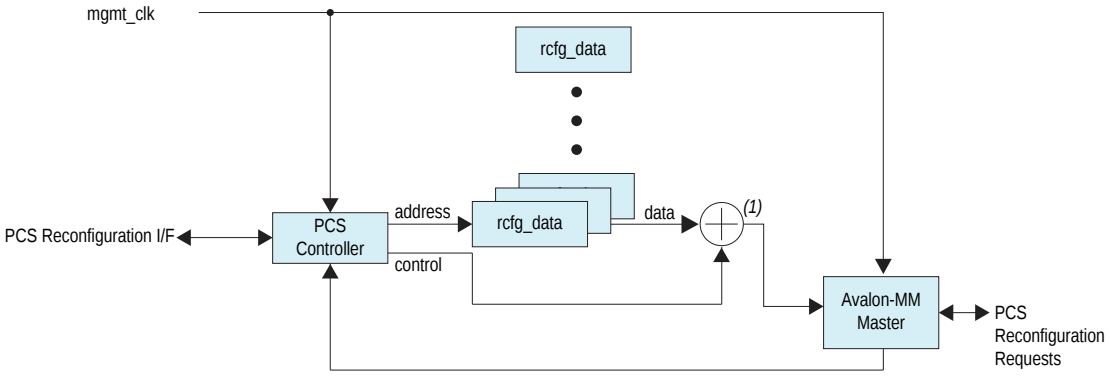
Reconfiguration Block

The Reconfiguration Block performs the Avalon-MM writes to the PHY for both PCS and PMA reconfiguration. The following figure shows reconfiguration blocks details. The Avalon-MM master accepts requests from the PMA or PCS controller. It performs the Read-Modify-Write or Write commands using the Avalon-MM interface. The PCS controller receives rate change requests from the Sequencer and translates them to a series of Read-Modify-Write or Write commands to the PMA and PCS.

Figure 2-60: Reconfiguration Block Diagram

Eight compile-time configurations are supported. These include the enumerated combination of reference clock (644 MHz or 322 MHz), including the 1588 mode, and the FEC sublayer

Figure 2-61: Reconfiguration Block Details



Note: Based on the control signal, the data is streamed to the AVMM master.

Related Information

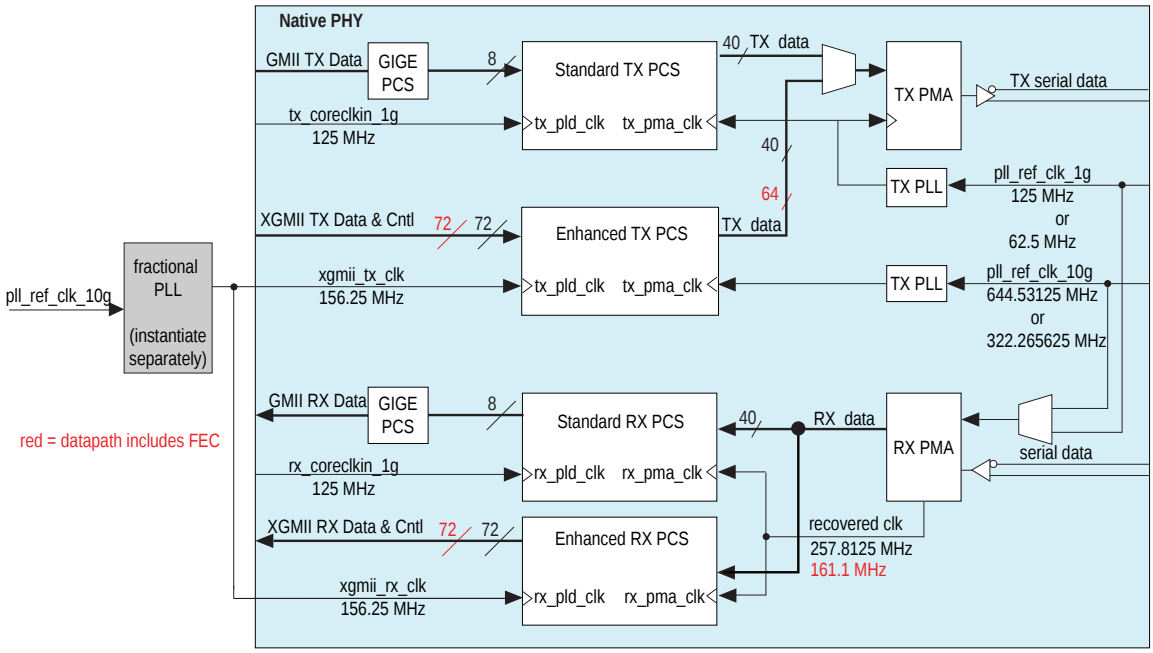
- **Arria 10 Enhanced PCS Architecture** on page 5-14
- **Arria 10 Standard PCS Architecture** on page 5-31

Clock and Reset Interfaces

You can use fPLL or CMU PLL to generate the clock for the TX PMA for the either the 1G or 10G data rate. For the 1G data rate, the frequency of the TX and RX clocks is 125 MHz, which is 1/8 of the MAC data rate. For 10G protocol, the frequency of TX and RX clocks is 156.25MHz, 1/64 of the MAC data rate. You can generate the 156.25MHz clock directly by using a fPLL, or you can divide the clock from TX PLL by 33. can be used. Bonded clocks are not supported for 1G/10GbE PHY.

The following figure provides an overview of the clocking for this core.

Figure 2-62: Clocks for Standard and 10G PCS and TX PLLs



The following table describes the clock and reset signals.

Table 2-106: Clock and Reset Signals

Signal Name	Direction	Description
rx_recovered_clk	Output	The RX clock which is recovered from the received data. You can use this clock as a reference to lock an external clock source. Its frequency is 125 or 156.25 MHz.
tx_clkout_1g	Output	GMII TX clock for the 1G TX parallel data source interface. The frequency is 125 MHz.
rx_clkout_1g	Output	GMII RX clock for the 1G RX parallel data source interface. The frequency is 125 MHz.
rx_coreclkin_1g	Input	Clock to drive the read side of the RX phase compensation FIFO in the Standard PCS. The frequency is 125 MHz.
tx_coreclkin_1g	Input	Clock to drive the write side of the TX phase compensation FIFO in the Standard PCS. The frequency is 125 MHz.
pll_ref_clk_1g	Input	Reference clock for the PMA block for the 1G mode. Its frequency is 125 or 62.5 MHz.
pll_ref_clk_10g	Input	Reference clock for the PMA block in 10G mode. Its frequency is 644.53125 or 322.265625 MHz.
pll_powerdown_1g	Input	Resets the 1Gb TX PLLs.

Signal Name	Direction	Description
pll_powerdown_10g	Input	Resets the 10Gb TX PLLs.
tx_analogreset	Input	Resets the analog TX portion of the transceiver PHY.
tx_digitalreset	Input	Resets the digital TX portion of the transceiver PHY.
rx_analogreset	Input	Resets the analog RX portion of the transceiver PHY.
rx_digitalreset	Input	Resets the digital RX portion of the transceiver PHY.
usr_an_lt_reset	Input	Resets only the AN and LT logic. This signal is only available for the 10GBASE-KR variants.
usr_seq_reset	Input	Resets the sequencer. Initiates a PCS reconfiguration, and may restart AN, LT or both if these modes are enabled.
usr_fec_reset	Input	When asserted, resets the 10GBASE-KR FEC module.
usr_soft_10g_pcs_reset	Input	When asserted, resets the 10G PCS associated with the FEC module.

Related Information

- **Input Reference Clock Sources** on page 3-24
- **PLLs** on page 3-3

Parameterizing the 1G/10GbE PHY

The Arria 10 1G/10GbE and 10GBASE-KR PHY IP Core allows you specify either the **Backplane-KR** or **1Gb/10Gb Ethernet** variant. When you select the **Backplane-KR** variant, the **Link Training (LT)** and **Auto Negotiation (AN)** tabs appear. The **1Gb/10Gb Ethernet** variant (1G/10GbE) does not implement LT and AN parameters.

Complete the following steps to configure the 1G/10GbE PHY IP Core in the MegaWizard Plug-In Manager:

1. For **Which device family will you be using?**, select Arria 10 from the list.
2. Click **Installed Plug-Ins** > **Interfaces** > **Ethernet**> **Arria 10 1G10GbE and 10BASE-KR PHY***ver*.
3. Select **Gb/10Gb Ethernet**from the **IP variant** list in MegaWizard Plug-In Manager.
4. Use the tabs on the MegaWizard Plug-In Manager to select the options required for the protocol.
5. Refer to the topics listed as Related Links to understand and specify 1G/10GbE parameters.
6. Click **Finish** to generate your parameterized 1G/10GbE PHY IP Core.

Related Information

- **General Options** on page 2-111
- **10GBASE-R Parameters** on page 2-111
- **10M/100M/1Gb Ethernet Parameters** on page 2-112
- **Speed Detection Parameters** on page 2-113
- **PHY Analog Parameters** on page 2-159

General Options

The General Options allow you to specify options common to 1GbE and 10GbE modes.

Table 2-107: General Options Parameters

Parameter Name	Options	Description
Initial datapath	**10G , 1G**	Specifies the data rate need after reset or power up. If you select **1G** for the initial datapath, the automatic speed detection function is not available.
Enable internal PCS reconfiguration logic	**On/Off**	When you turn this option on, the core includes reconfiguration logic to dynamically change the initial configuration.
Enable IEEE 1588 Precision Time Protocol	**On/Off**	When you turn this option on, the core includes the soft FIFO for 1588, the associated logic, and enables the ports required for IEEE 1588 PTP.
Enable tx_pma_clkout port	**On/Off**	When you turn this option on, the `tx_pma_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_pma_clkout port	**On/Off**	When you turn this option on, the `rx_pma_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable tx_divclk port	**On/Off**	When you turn this option on, the `tx_divclk` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_divclk port	**On/Off**	When you turn this option on, the `rx_divclk` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable tx_clkout port	**On/Off**	When you turn this option on, the `tx_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.
Enable rx_clkout port	**On/Off**	When you turn this option on, the `rx_clkout` port is enabled. Refer to clock and reset signals section for more information about his port.

10GBASE-R Parameters

The 10GBASE-R parameters specify basic features of the 10GBASE-R PCS. The FEC options also allow you to specify the FEC ability.

Table 2-108: 10GBASE-R Parameters

Parameter Name	Options	Description
Reference clock frequency	**644.53125 MHz** **322.265625**	Specifies the input reference clock frequency. The default is **322.265625MHz**.
Enable additional control and status pins	**On/Off**	When you turn this option on, the core includes the `rx_block_lock` and `rx_hi_ber` ports.

Table 2-109: FEC Options

Parameter Name	Options	Description
Include FEC sublayer	**On/Off**	When you turn this option on, the core includes logic to implement FEC and a soft 10GBASE-R PCS.
Set `FEC_ability` bit on power up and reset	**On/Off**	When you turn this option on, the core sets the `Assert KR FEC Ability` bit (0xB0[16]) FEC ability bit during power up and reset, causing the core to advertise the FEC ability. This option is required for FEC functionality.
Set `FEC_Enable` bit on power up and reset	**On/Off**	When you turn this option **On**, the core sets the `KR FEC Request` bit (0xB0[18]) during power up and reset, causing the core to request the FEC ability during Auto Negotiation. This option is required for FEC functionality.

10M/100M/1Gb Ethernet Parameters

The 10M/100M/1GbE parameters allow you to specify options for the MII interface and the 1GbE data rate.

Table 2-110: 10M/100M/1Gb Ethernet

Parameter Name	Options	Description
Enable 1Gb Ethernet protocol	**On/Off**	When you turn this option on, the core includes the GMII interface and related logic.
Enable 10M/100Mb Ethernet functionality	**On/Off**	When you turn this option on, the core includes the MII PCS. It also supports 4-speed mode to implement a 10M/100M interface to the MAC for the GbE line rate.

Parameter Name	Options	Description
PHY ID (32 bits)	`32-bit value`	An optional 32-bit value that serves as a unique identifier for a particular type of PCS. The identifier includes the following components: • Bits 3-24 of the Organizationally Unique Identifier (OUI) assigned by the IEEE • 6-bit model number • 4-bit revision number If unused, do not change the default value which is 0x00000000.
PHY Core version (16 bits)	`16-bit value`	This is an optional 16-bit value identifies the PHY core version.

Speed Detection Parameters

By selecting the **Enable automatic speed detection** option in the MegaWizard Plug-In Manager, the PHY IP includes the sequencer module which implements the Parallel Detect function as described in the Ethernet specification.

Selecting the speed detection option gives the PHY the ability to detect to link partners that support 1G/10GbE but have disabled Auto-Negotiation. If you turn on the **Enable automatic speed detection** parameter, the PHY includes the sequencer block. During Auto-Negotiation, if AN cannot detect Differential Manchester Encoding (DME) pages from link partner, the Sequencer reconfigures to 1GE and 10GE modes (Speed/Parallel detection) until it detects a valid 1G or 10GbE pattern.

Table 2-111: Speed Detection

Parameter Name	Options	Description
Enable automatic speed detection	**On/Off**	When you turn this option **On**, the core includes the Sequencer block that sends reconfiguration requests to detect 1G or 10GbE when the Auto Negotiation block is not able detect AN data.
Avalon-MM clock frequency	`100-125 MHz`	Specifies the clock frequency for `phy_mgmt_clk`.
Link fail inhibit time for 10Gb Ethernet	`504 ms`	Specifies the time before `link_status` is set to FAIL or OK. A link fails if the `link_fail_inhibit_time` has expired before `link_status` is set to OK. The legal range is 500-510 ms. For more information, refer to *"Clause 73 Auto-Negotiation for Backplane Ethernet"* in *IEEE Std 802.3ap-2007*.
Link fail inhibit time for 1Gb Ethernet	`40-50 ms`	Specifies the time before `link_status` is set to FAIL or OK . A link fails if the link_fail_inhibit_time has expired before `link_status` is set to OK. The legal range is 40-50 ms. For more information, refer to *"Clause 73 Auto-Negotiation for Backplane Ethernet"* in *IEEE Std 802.3ap-2007*.

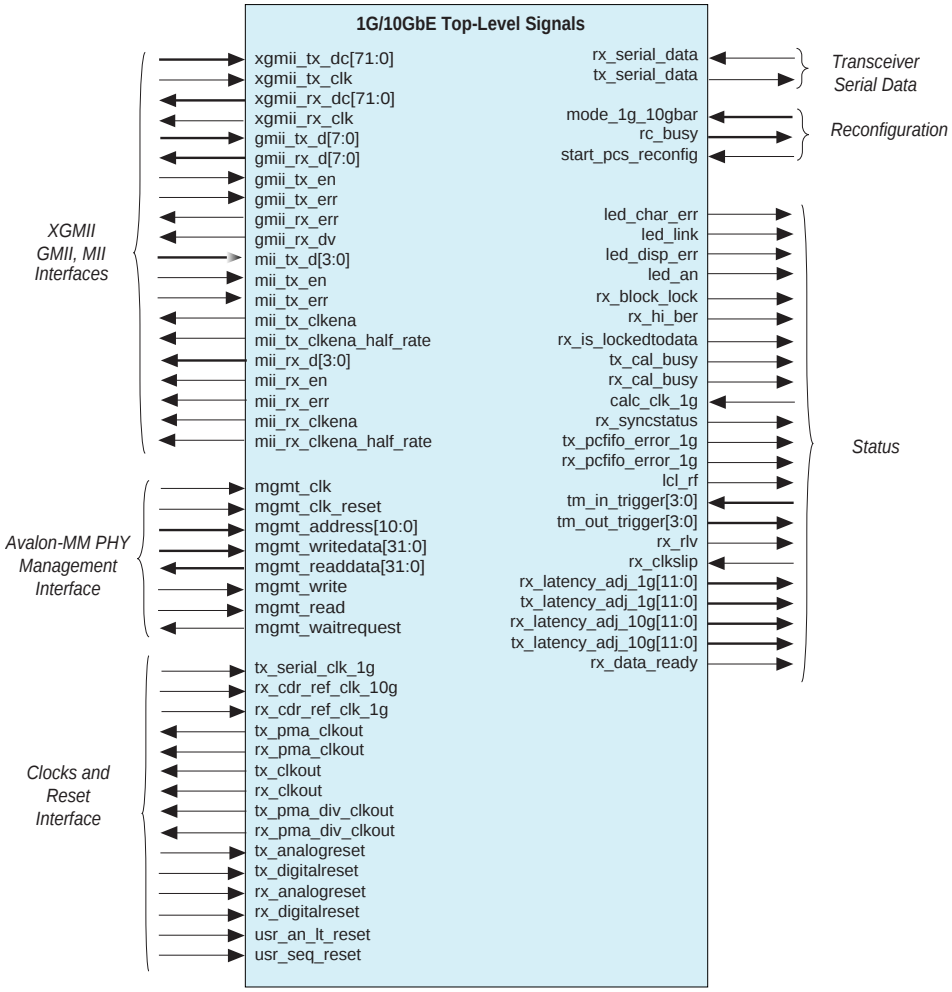
PHY Analog Parameters

You can specify analog parameters using the Quartus II Assignment Editor, the Pin Planner, or through the Quartus II Settings File (**.qsf**).

Refer to the appropriate link for a description of analog parameters.

1G/10GbE PHY Interfaces

Figure 2-63: 1G/10GbE PHY Top-Level Signals



The block diagram shown in the GUI labels the external pins with the interface type and places the interface name inside the box. The interface type and name are used in the **_hw.tcl** file. If you turn on **Show signals**, the **block diagram** displays all top-level signal names. For more information about **_hw.tcl** files, refer to refer to the *Component Interface Tcl Reference* chapter in volume 1 of the *Quartus II Handbook*

Note: Some of the signals shown in are this figure are unused and will be removed in a future release. The descriptions of these identifies them as not functional.

Related Information

Component Interface Tcl Reference

Clock and Reset Interfaces

This topic defines the clock and reset signals.

The following table describes the clock and reset signals.

Table 2-112: Clock and Reset Signals

Signal Name	Direction	Description
`tx_serial_clk_10g`	Input	High speed clock from the 10G PLL to drive 10G PHY TX PMA. The frequency of this clock is 5.15625 GHz.
`tx_serial_clk_1g`	Input	High Speed clock from 1G PLL to drive the 1G PHY TX PMA. This clock is not required if GbE is not used. The frequency of this clock is 500 MHz.
`rx_cdr_ref_clk_10g`	Input	10G PHY RX PLL reference clock . This clock frequency can be 644.53125 MHz or 322.2656 MHz.
`rx_cdr_ref_clk_1g`	Input	1G PHY RX PLL reference clock. The frequency is 125 MHz. This clock is only required if 1G is enabled.
`tx_pma_clkout`	Output	This clock is used for the 1588 mode TX soft FIFO and 1G TX PCS parallel data. This clock frequency is 125 MHz for 1G and 257.81 MHz for 10G. This clock frequency is 161.13 MHz for 10G with FEC enabled.
`rx_pma_clkout`	Output	This clock is used for the 1588 mode RX soft FIFO and 1G RX PCS parallel data. This clock frequency is 125 MHz for 1G and 257.81 MHz for 10G. This clock frequency is 161.13 MHz for 10G with FEC enabled.
`tx_clkout`	Output	XGMII/GMII TX clock for the TX parallel data source interface. This clock frequency is 125 MHz in 1G mode and 257.81 MHz in 10G Mode, and 161.13 MHz with FEC enabled.
`rx_clkout`	Output	XGMII/GMII RX clock for the RX parallel data source interface. This clock frequency is 125 MHz in 1G mode and 257.81 in 10G Mode, and 161.13 MHz with FEC enabled.
`tx_pma_div_clkout`	Output	This is the divided 33 clock from the TX serializer. You can use this clock for the for `xgmii_tx_clk` or `xgmii_rx_clk`. This clock frequency is 125 MHz for 1G and 156.25 MHz for 10G. The frequencies are the same if you enable 1588 or FEC.
`rx_pma_div_clkout`	Output	This is the divided 33 clock from CDR recovered clock. This clock frequency is 125 MHz for 1G and 156.25 MHz for 10G. The frequencies are the same if you enable 1588 or FEC. This clock is not used for clock the 10G RX datapath. If PHY is reconfigured to 1G mode, this clock frequency changes to 125 MHz.
`tx_analogreset`	Input	Resets the analog TX portion of the transceiver PHY.
`tx_digitalreset`	Input	Resets the digital TX portion of the transceiver PHY.
`rx_analogreset`	Input	Resets the analog RX portion of the transceiver PHY.

Signal Name	Direction	Description
rx_digitalreset	Input	Resets the digital RX portion of the transceiver PHY.
usr_seq_reset	Input	Resets the sequencer. Initiates a PCS reconfiguration, and may restart AN, LT or both if these modes are enabled.

Related Information

- **Input Reference Clock Sources** on page 3-24
- **PLLs** on page 3-3

Data Interfaces

The following table describes the signals in the XGMII. GMII, and MII interfaces. The MAC drives the TX XGMII, GMII, and MII signals to the 1G/10GbE PHY. The 1G/10GbE PHY drives the RX XGMII, GMII, or MII signals to the MAC.

Table 2-113: SGMII and GMII Signals

Signal Name	Direction	Clock Domain	Description
1G/10GbE XGMII Data Interface			
xgmii_tx_dc[71:0]	Input	Synchronous to xgmii_tx_clk	XGMII data and control for 8 lanes. Each lane consists of 8 bits of data and 1 bit of control.
xgmii_tx_clk	Input	Clock signal	Clock for single data rate (SDR) XGMII TX interface to the MAC. It should connect to xgmii_rx_clk. This clock can be connected to the tx_div_clkout; however, Altera recommends that you connect it to a PLl for use with the Triple Speed Ethernet MegaCore Function. The frequency is 125 MHz for 1G and 156.25 MHz for 10G. This clock is driven from the MAC. The frequencies are the same if you enable 1588 or FEC.
xgmii_rx_dc[71:0]	Output	Synchronous to xgmii_rx_clk	RX XGMII data and control for 8 lanes. Each lane consists of 8 bits of data and 1 bit of control.
xgmii_rx_clk	Input	Clock signal	Clock for SDR XGMII RX interface to the MAC. This clock can be connected to the tx_div_clkout ; however, Altera recommends that you connect it to a PLl for use with the Triple Speed Ethernet MegaCore Function. The frequency is 125 MHz for 1G and 156.25 MHz for 10G. This clock is driven from the MAC. The frequencies are the same if you enable 1588 or FEC.
1G/10GbE GMII Data Interface			

Signal Name	Direction	Clock Domain	Description
gmii_tx_d[7:0]	Input	Synchronous to tx_pma_clkout	TX data for 1G mode. Synchronized to tx_pma_clkout clock. The TX PCS 8B/10B module encodes this data which is sent to link partner.
gmii_rx_d[7:0]	Output	Synchronous to rx_pma_clkout	RX data for 1G mode. Synchronized to rx_pma_clkout clock. The RX PCS 8B/10B decoders decodes this data and sends it to the MAC.
gmii_tx_en	Input	Synchronous to tx_pma_clkout	When asserted, indicates the start of a new frame. It should remain asserted until the last byte of data on the frame is present on gmii_tx_d.
gmii_tx_err	Input	Synchronous to tx_pma_clkout	When asserted, indicates an error. May be asserted at any time during a frame transfer to indicate an error in that frame.
gmii_rx_err	Output	Synchronous to rx_pma_clkout	When asserted, indicates an error. May be asserted at any time during a frame transfer to indicate an error in that frame.
gmii_rx_dv	Output	Synchronous to rx_pma_clkout	When asserted, indicates the start of a new frame. It remains asserted until the last byte of data on the frame is present on gmii_rx_d.
MII Data Interface			
mii_tx_d[3:0]	Input	Synchronous to tx_pma_clkout	TX data to be encoded and sent to the link partner. Synchronized to the tx_pma_clkout clock.
mii_tx_en	Input	Synchronous to tx_pma_clkout	When asserted, indicates the start of a new frame. mii_tx_enshould remain asserted until the last nibble of data on the frame is present on mii_tx_d[3:0].
mii_tx_err	Input	Synchronous to tx_pma_clkout	When asserted, it indicates an error in the frame. mii_tx_en should also be asserted for PHY to transmit invalid data.
mii_tx_clkena	Output	Clock signal	MII TX clock enable. This clock frequency has the following frequencies: • 25 MHz: For an effective rate of 100 Mbps • 2.5 MHz: For an effective rate of 10 Mbps
mii_tx_clkena_half_rate	Output	Clock signal	MII RX clock enable when the FPGA fabric runs at half the PCS frequency. This clock frequency has the following frequencies: • 12.5 MHz: For an effective rate of 100 Mbps • 1.25 MHz: For an effective rate of 10 Mbps

Signal Name	Direction	Clock Domain	Description
mii_rx_d[3:0]	Output	Synchronous to rx_pma_clkout	RX Data received from link partner. Synchronized to the rx_pma_clkout clock.
mii_rx_en	Output	Synchronous to rx_pma_clkout	When asserted, indicates that data onmii_rx_d[3:0] is valid.
mii_rx_err	Output	Synchronous to rx_pma_clkout	When asserted, it indicates an error in the frame.
mii_rx_clkena	Output	Clock signal	MII RX clock enable. This clock frequency has the following frequencies: • 25 MHz: For an effective rate of 100 Mbps • 2.5 MHz: For an effective rate of 10 Mbps
mii_rx_clkena_half_rate	Output	Clock signal	MII RX clock enable when the FPGA fabric runs at half the PCS frequency. This clock frequency has the following frequencies: • 12.5 MHz: For an effective rate of 100 Mbps • 1.25 MHz: For an effective rate of 10 Mbps
mii_speed_sel[1:0]	Output	Asynchronous signal	Specifies the SGMII (4-speed) mode. The following encodings are defined: • 2b'00: 10 Gbps • 2'b01: 1 Gbps • 2'b10: 100 Mbps • 2'b11: 10 Mbps

XGMII Mapping to Standard SDR XGMII Data

The 72-bit TX XGMII data bus format is different than the standard SDR XGMII interface. The following table shows the mapping of this non-standard format to the standard SDR XGMII interface

Table 2-114: TX XGMII Mapping to Standard SDR XGMII Interface

Signal Name	SDR XGMII Signal Name	Description
xgmii_tx_dc[7:0]	xgmii_sdr_data[7:0]	Lane 0 data
xgmii_tx_dc[8]	xgmii_sdr_ctrl[0]	Lane 0 control
xgmii_tx_dc[16:9]	xgmii_sdr_data[15:8]	Lane 1 data
xgmii_tx_dc[17]	xgmii_sdr_ctrl[1]	Lane 1 control
xgmii_tx_dc[25:18]	xgmii_sdr_data[23:16]	Lane 2 data

Signal Name	SDR XGMII Signal Name	Description
xgmii_tx_dc[26]	xgmii_sdr_ctrl[2]	Lane 2 control
xgmii_tx_dc[34:27]	xgmii_sdr_data[31:24]	Lane 3 data
xgmii_tx_dc[35]	xgmii_sdr_ctrl[3]	Lane 3 control
xgmii_tx_dc[43:36]	xgmii_sdr_data[39:32]	Lane 4 data
xgmii_tx_dc[44]	xgmii_sdr_ctrl[4]	Lane 4 control
xgmii_tx_dc[52:45]	xgmii_sdr_data[47:40]	Lane 5 data
xgmii_tx_dc[53]	xgmii_sdr_ctrl[5]	Lane 5 control
xgmii_tx_dc[61:54]	xgmii_sdr_data[55:48]	Lane 6 data
xgmii_tx_dc[62]	xgmii_sdr_ctrl[6]	Lane 6 control
xgmii_tx_dc[70:63]	xgmii_sdr_data[63:56]	Lane 7 data
xgmii_tx_dc[71]	xgmii_sdr_ctrl[7]	Lane 7 control

The 72-bit RX XGMII data bus format is different from the standard SDR XGMII interface. The following table shows the mapping of this non-standard format to the standard SDR XGMII interface:

Table 2-115: RX XGMII Mapping to Standard SDR XGMII Interface

Signal Name	XGMII Signal Name	Description
xgmii_rx_dc[7:0]	xgmii_sdr_data[7:0]	Lane 0 data
xgmii_rx_dc[8]	xgmii_sdr_ctrl[0]	Lane 0 control
xgmii_rx_dc[16:9]	xgmii_sdr_data[15:8]	Lane 1 data
xgmii_rx_dc[17]	xgmii_sdr_ctrl[1]	Lane 1 control
xgmii_rx_dc[25:18]	xgmii_sdr_data[23:16]	Lane 2 data
xgmii_rx_dc[26]	xgmii_sdr_ctrl[2]	Lane 2 control
xgmii_rx_dc[34:27]	xgmii_sdr_data[31:24]	Lane 3 data
xgmii_rx_dc[35]	xgmii_sdr_ctrl[3]	Lane 3 control
xgmii_rx_dc[43:36]	xgmii_sdr_data[39:32]	Lane 4 data
xgmii_rx_dc[44]	xgmii_sdr_ctrl[4]	Lane 4 control
xgmii_rx_dc[52:45]	xgmii_sdr_data[47:40]	Lane 5 data

Signal Name	XGMII Signal Name	Description
xgmii_rx_dc[53]	xgmii_sdr_ctrl[5]	Lane 5 control
xgmii_rx_dc[61:54]	xgmii_sdr_data[55:48]	Lane 6 data
xgmii_rx_dc[62]	xgmii_sdr_ctrl[6]	Lane 6 control
xgmii_rx_dc[70:63]	xgmii_sdr_data[63:56]	Lane 7 data
xgmii_rx_dc[71]	xgmii_sdr_ctrl[7]	Lane 7 control

Serial Data Interface

This topic describes the serial data interface.

Signal Name	Direction	Description
rx_serial_data	Input	RX serial input data
tx_serial_data	Output	TX serial output data

Control and Status Interfaces

The XGMII and GMII interface signals drive data to and from PHY.

Table 2-116: Control and Status Signals

Signal Name	Direction	Clock Domain	Description
led_char_err	Output	Synchronous to rx_clkout	10-bit character error. Asserted for one rx_clkout_1g cycle when an erroneous 10-bit character is detected.
led_link	Output	Synchronous to rx_clkout	When asserted, indicates successful link synchronization.
led_disp_err	Output	Synchronous to rx_clkout	Disparity error signal indicating a 10-bit running disparity error. Asserted for one rx_clkout_1g cycle when a disparity error is detected. A running disparity error indicates that more than the previous and perhaps the current received group had an error.
led_an	Output	Synchronous to rx_clkout	Clause 37 Auto-negotiation status. The PCS function asserts this signal when auto-negotiation completes.
rx_block_lock	Output	Synchronous to rx_clkout	Asserted to indicate that the block synchronizer has established synchronization.
rx_hi_ber	Output	Synchronous to rx_clkout	Asserted by the BER monitor block to indicate a Sync Header high bit error rate greater than 10^{-4}.
rx_is_lockedtodata	Output	Asynchronous signal	When asserted, indicates the RX channel is locked to input data.
tx_cal_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that the TX channel is being calibrated.

Signal Name	Direction	Clock Domain	Description
rx_cal_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that the RX channel is being calibrated.
calc_clk_1g	Input	Clock signal	This clock is used to calculate the latency of the soft 1G PCS block. This clock is only required for when you enable 1588 in 1G mode. Its frequency is 80 MHz. This clock should have the same PPM as the pll_ref_clk_1ginput.
rx_sync_status	Output	Synchronous to rx_clkout	When asserted, indicates the word aligner has aligned to in incoming word alignment pattern.
tx_pcfifo_error_1g	Output	Synchronous to tx_clkout	When asserted, indicates that the Standard PCS TX phase compensation FIFO is full.
rx_pcfifo_error_1g	Output	Synchronous to rx_clkout	When asserted, indicates that the Standard PCS RX phase compensation FIFO is full.
lcl_rf	Input	Synchronous to xgmii_tx_clk	When asserted, indicates a Remote Fault (RF).The MAC sends this fault signal to its link partner. Bit D13 of the Auto Negotiation Advanced Remote Fault register (0xC2) records this error.
rx_clkslip	Input	Asynchronous signal	When asserted, indicates that the deserializer has either skipped one serial bit or paused the serial clock for one cycle to achieve word alignment. As a result, the period of the parallel clock could be extended by 1 unit interval (UI) during the clock slip operation.
rx_latency_adj_1g[11:0]	Output	Synchronous to rx_clkout	When you enable 1588, this signal outputs the real time latency in GMII clock cycles (125 MHz) for the RX PCS and PMA datapath for 1G mode.
tx_latency_adj_1g[11:0]	Output	Synchronous to tx_clkout	When you enable 1588, this signal outputs real time latency in GMII clock cycles (125 MHz) for the TX PCS and PMA datapath for 1G mode.
rx_latency_adj_10g[11:0]	Output	Synchronous to rx_clkout	When you enable 1588, this signal outputs the real time latency in XGMII clock cycles (156.25 MHz) for the RX PCS and PMA datapath for 10G mode.
tx_latency_adj_10g[11:0]	Output	Synchronous to tx_clkout	When you enable 1588, this signal outputs real time latency in XGMII clock cycles (156.25 MHz) for the TX PCS and PMA datapath for 10G mode.
rx_data_ready	Output	Synchronous to rx_clkout	When asserted, indicates that the MAC can begin sending data to the PHY.

Dynamic Reconfiguration Interface

You can use the dynamic reconfiguration interface signals to dynamically change between 1G and 10G data rates.

Table 2-117: Dynamic Reconfiguration Interface Signals

Signal Name	Direction	Clock Domain	Description
mode_1g_10gbar	Input	Synchronous to mgmt_clk	This signal indicates the requested mode for the channel. A 1 indicates 1G mode and a 0 indicates 10G mode.
rc_busy	Output	Synchronous to mgmt_clk	When asserted, indicates that reconfiguration is in progress. Synchronous to the mgmt_clk. This signal is only exposed under the following conditions: • Turn off **Enable automatic speed detection** • Turn on **Enable internal PCS reconfiguration logic** • Turn on **Enable 1Gb Ethernet protocol**
start_pcs_reconfig	Input	Synchronous to mgmt_clk	When asserted, initiates reconfiguration of the PCS. Sampled with the mgmt_clk. This signal is only exposed under the following conditions: • Turn off **Enable automatic speed detection** • Turn on **Enable internal PCS reconfiguration logic** • Turn on **Enable 1Gb Ethernet protocol**

Avalon-MM Register Interface

The Avalon-MM slave interface signals provide access to all registers.

Table 2-118: Avalon-MM Interface Signals

Signal Name	Direction	Clock Domain	Description
mgmt_clk	Input	Clock	The clock signal that controls the Avalon-MM PHY management, interface. If you plan to use the same clock for the PHY management interface and transceiver reconfiguration, you must restrict the frequency range to 100-125 MHz to meet the specification for the transceiver reconfiguration clock.
mgmt_clk_reset	Input	Reset	Resets the PHY management interface. This signal is active high and level sensitive.
mgmt_addr[10:0]	Input	Synchronous to mgmt_clk	11-bit Avalon-MM address.
mgmt_writedata[31:0]	Input	Synchronous to mgmt_clk	Input data.
mgmt_readdata[31:0]	Output	Synchronous to mgmt_clk	Output data.
mgmt_write	Input	Synchronous to mgmt_clk	Write signal. Active high.
mgmt_read	Input	Synchronous to mgmt_clk	Read signal. Active high.

Signal Name	Direction	Clock Domain	Description
mgmt_waitrequest	Output	Synchronous to mgmt_clk	When asserted, indicates that the Avalon-MM slave interface is unable to respond to a read or write request. When asserted, control signals to the Avalon-MM slave interface must remain constant.

Related Information

Avalon Interface Specifications

Register Definitions

The Avalon-MM master interface signals provide access to the control and status registers.

The following table specifies the control and status registers that you can access over the Avalon-MM PHY management interface. A single address space provides access to all registers.

Note: Unless otherwise indicated, the default value of all registers is 0.

Note: Writing to reserved or undefined register addresses may have undefined side effects.

Table 2-119: 10GBASE-KR Register Definitions

Word Addr	Bit Offset	R/W	Name	Description
0x4B0	0	RW	`Reset SEQ`	When set to 1, resets the 10GBASE-KR sequencer, initiates a PCS reconfiguration, and may restart Auto-Negotiation, Link Training or both if AN and LT are enabled (10GBASE-KR mode). SEQ Force Mode[2:0] forces these modes. This reset self clears.
	1	RW	`Disable AN Timer`	Auto-Negotiation disable timer. If disabled (`Disable AN Timer = 1`) , AN may get stuck and require software support to remove the ABILITY_DETECT capability if the link partner does not include this feature. In addition, software may have to take the link out of loopback mode if the link is stuck in the ACKNOWLEDGE_DETECT state. To enable this timer set `Disable AN Timer = 0`.
	2	RW	`Disable LF Timer`	When set to 1, disables the Link Fault timer. When set to 0, the Link Fault timer is enabled.
	6:4	RW	`SEQ Force Mode[2:0]`	Forces the sequencer to a specific protocol. Must write the `Reset SEQ` bit to 1 for the Force to take effect. The following encodings are defined: • 3'b000: No force • 3'b001: GigE • 3'b010: Reserved • 3'b011: Reserved • 3'b100: 10GBASE-R • 3'b101: 10GBASE-KR • Others: Reserved
	16	RW	`Assert KR FEC Ability`	When set to 1, indicates that the FEC ability is supported. This bit defaults to 1 if the **Set FEC_ability bit on power up/reset** bit is on. For more information, refer to the FEC variable `FEC_Enable` as defined in *Clause 74.8.2* and 10GBASE-KR PMD control register bit (1.171.0) IEEE 802.3ap-2007.
	18	RW	`Assert KR FEC Request`	When set to 1, indicates that the core is requesting the FEC ability. When this bit changes, you must assert the `Reset SEQ` bit (0x4B0[0]) to renegotiate with the new value.

Word Addr	Bit Offset	R/W	Name	Description
0x4B1	0	R	SEQ Link Ready	When asserted, the sequencer is indicating that the link is ready.
	1	R	SEQ AN timeout	When asserted, the sequencer has had an Auto-Negotiation timeout. This bit is latched and is reset when the sequencer restarts Auto-Negotiation.
	2		SEQ LT timeout	When set, indicates that the Sequencer has had a timeout.
	13:8		SEQ Reconfig Mode[5:0]	Specifies the Sequencer mode for PCS reconfiguration. The following modes are defined: • Bit 8, mode[0]: AN mode • Bit 9, mode[1]: LT Mode • Bit 10, mode[2]: 10G data mode • Bit 11, mode[3]: Gige data mode • Bit 12, mode[4]: Reserved for XAUI • Bit13, mode[5]: 10G FEC mode
	16	R	KR FEC Ability	Indicates whether or not the 10GBASE-KR PHY supports FEC. For more information, refer to the FEC variable FEC_Enable as defined in *Clause 74.8.2* and 10GBASE-KR PMD control register bit (1.171.0) IEEE 802.3ap-2007.
0x4B2	10:0		Reserved	—
	11	RWSC	FEC TX Error Insert	Writing a 1 inserts 1 error pulse into the TX FEC depending on the Transcoder and Burst error settings. Software clears this register.
	31:15	RWSC	Reserved	—
0x0DC 0x0DD 0x0DE 0x0DF	7:0	RSC	FEC Corrected Blocks	Counts the number of corrected FEC blocks. Resets to 0 when read. Otherwise, it holds at the maximum count and does not roll over. Refer to *Clause 74.8.4.1 of IEEE 802.3ap-2000* for details. The low-order byte of each register maps to the following bits of the 32 bit counter: • 0x0DC[7:0]: [7:0] • 0x0DD[7:0]: [15:8] • 0x0DE[7:0]: [23:16] • 0x0DF[7:0]:[31:24]
0x0E0 0x0E1 0x0E2 0x0E3	7:0	RSC	FEC Uncorrected Blocks	Counts the number of uncorrectable FEC blocks. Resets to 0 when read. Otherwise, it holds at the maximum count and does not roll over. Refer to *Clause 74.8.4.1 of IEEE 802.3ap-2000* for details. The low-order byte of each register maps to the following bits of the 32 bit counter: • 0x0E0[7:0]: [7:0] • 0x0E0[7:0]: [15:8] • 0x0E0[7:0]: [23:16] • 0x0E0[7:0]: [31:24]

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 6-1

Hard Transceiver PHY Registers

Table 2-120: Hard Transceiver PHY Registers

Addr	Bit	Access	Name	Description
0x000-0x3FF	[9:0]	RW	Access to HSSI registers	All registers in the PCS and PMA that you can dynamically reconfigure are in this address space. Refer to reconfiguration chapter for further information.

Enhanced PCS Registers

These registers provide Enhanced PCS status information.

Table 2-121: PCS Registers

Addr	Bit	Access	Name	Description
0x480	31:0	RW	Indirect_addr	Because the PHY implements a single channel, this register must remain at the default value of 0 to specify logical channel 0.
0x481	2	RW	RCLR_ERRBLK_CNT	Error Block Counter clear register. When set to 1, clears the RCLR_ERRBLK_CNT register. When set to 0, normal operation continues.
	3	RW	RCLR_BER_COUNT	BER Counter clear register. When set to 1, clears the RCLR_BER_COUNT register. When set to 0, normal operation continues.
0x482	1	RO	HI_BER	High BER status. When set to 1, the PCS is reporting a high BER. When set to 0, the PCS is not reporting a high BER.
	2	RO	BLOCK_LOCK	Block lock status. When set to 1, the PCS is locked to received blocks. When set to 0, the PCS is not locked to received blocks.
	3	RO	TX_FULL	When set to 1, the TX_FIFO is full.
	4	RO	RX_FULL	When set to 1, the RX_FIFO is full.
	5	RO	RX_SYNC_HEAD_ERROR	When set to 1, indicates an RX synchronization error.
	6	RO	RX_SCRAMBLER_ERROR	When set to 1, indicates an RX scrambler error.
	7	RO	Rx_DATA_READY	When set to 1, indicates the PHY is ready to receive data.

Arria 10 GMII PCS Registers

This topic describes the GMII PCS registers.

Addr	Bit	R/W	Name	Description
0x490	9	RW	RESTART_AUTO_NEGOTIATION	Set this bit to 1 to restart the Clause 37 Auto-Negotiation sequence. For normal operation, set this bit to 0 which is the default value. This bit is self-clearing.
	12	RW	AUTO_NEGOTIATION_ENABLE	Set this bit to 1 to enable Clause 37 Auto-Negotiation. The default value is 1.
	15	RW	Reset	Set this bit to 1 to generate a synchronous reset pulse which resets all the PCS state machines, comma detection function, and the 8B/10B encoder and decoder. For normal operation, set this bit to 0. This bit self clears.
0x491	2	R	LINK_STATUS	A value of 1 indicates that a valid link is operating. A value of 0 indicates an invalid link. If link synchronization is lost, this bit is 0.
	3	R	AUTO_NEGOTIATION_ABILITY	A value of 1 indicates that the PCS function supports Clause 37 Auto-Negotiation.
	5	R	AUTO_NEGOTIATION_COMPLETE	A value of 1 indicates the following status: • The Auto-Negotiation process is complete. • The Auto-Negotiation control registers are valid.
0x494	5	RW	FD	Full-duplex mode enable for the local device. Set to 1 for full-duplex support.
	6	RW	HD	Half-duplex mode enable for the local device. Set to 1 for half-duplex support. This bit should always be set to 0.
	8:7	RW	PS2,PS1	Pause support for local device. The following encodings are defined for PS1/PS2: • 2'b00: Pause is not supported • 2'b0 1: Asymmetric pause toward link partner • 2'b10: Symmetric pause • 2'b11: Pause is supported on TX and RX
	13:12	RW	RF2,RF1	Remote fault condition for local device. The following encodings are defined for RF1/RF2: • 2'b00: No error, link is valid (reset condition) • 2'b0 1: Offline • 2'b10: Failure condition • 2'b11: Auto-negotiation error
	14	R0	ACK	Acknowledge for local device. A value of 1 indicates that the device has received three consecutive matching ability values from its link partner.
	15	RW	NP	Next page. In the device ability register, this bit is always set to 0.

Addr	Bit	R/W	Name	Description
0x495	5	R	FD	Full-duplex mode enable for the link partner. This bit should always be 1 because only full duplex is supported.
	6	R	HD	Half-duplex mode enable for the link partner. A value of 1 indicates support for half duplex. This bit should always be 0 because half-duplex mode is not supported.
	8:7	R	PS2,PS1	Specifies pause support for link partner. The following encodings are defined for PS1/PS2: • 2'b00: Pause is not supported • 2'b0 1: Asymmetric pause toward link partner • 2'b10: Symmetric pause • 2'b11: Pause is supported on TX and RX
	13:12	R	RF2,RF1	Remote fault condition for link partner. The following encodings are defined for RF1/RF2: • 2'b00: No error, link is valid (reset condition) • 2'b0 1: Offline • 2'b10: Failure condition • 2'b11: Auto-negotiation error
	14	R	ACK	Acknowledge for link partner. A value of 1 indicates that the device has received three consecutive matching ability values from its link partner.
	15	R	NP	Next page. In link partner register. When set to 0, the link partner has a Next Page to send. When set to 1, the link partner does not a Next Page. Next Page is not supported in Auto Negotiation.
0x496	0	R	LINK_ PARTNER_ AUTO_ NEGOTIATION_ ABLE	Set set to 1, indicates that the link partner supports auto negotiation. The default value is 0.
	1	R	PAGE_RECEIVE	A value of 1 indicates that a new page has been received with new partner ability available in the register partner ability. The default value is 0 when the system management agent performs a read access.
0x4A2	15:0	RW	AN link timer[15:0]	Low-order 16 bits of the 21-bit auto-negotiation link timer. Each timer step corresponds to 8ns (assuming a 125 MHz clock). The total timer corresponds to 16 ms. The reset value sets the timer to 10 ms for hardware mode and 10 us for simulation mode.
0x4A3	4:0	RW	AN link timer[4:0]	High-order 5 bits of the 21-bit auto-negotiation link timer.

Addr	Bit	R/W	Name	Description
0x4A4	0	RW	SGMII_ENA	Determines the PCS function operating mode. Setting this bit to 1b'1 enables SGMII mode. Setting this bit to 1b'0 enables 1000BASE-X gigabit mode.
	1	RW	USE_SGMII_AN	In SGMII mode, setting this bit to 1b'1 causes the PCS to be configured with the link partner abilities advertised during auto-negotiation. If this bit is set to 1b'0, the PCS function should be configured with the SGMII_SPEED and SGMII_DUPLEX bits.
	3:2	RW	SGMII_SPEED	SGMII speed. When the PCS operates in SGMII mode (SGMII_ENA = 1) and is not programmed for automatic configuration (USE_SGMII_AN = 0), the following encodings specify the speed : • 2'b00: 10 Mbps • 2'b01: 100 Mbps • 2'b10: Gigabit • 2'b11: Reserved These bits are not used when SGMII_ENA = 0or USE_SGMII_AN = 1.

PMA Registers

The PMA registers allow you to reset the PMA, customize the TX and RX serial data interface, and provide status information.

Address	Bit	R/W	Name	Description
0x422	[<p>-1:0]	RO	pma_tx_pll_is_locked	Indicates that the TX PLL is locked to the input reference clock. <p> is the number of PLLs.
0x444	1	RW	reset_tx_digital	Writing a 1 causes the internal TX digital reset signal to be asserted. You must write a 0 to clear the reset condition.
	2	RW	reset_rx_analog	Writing a 1 causes the internal RX analog reset signal to be asserted. You must write a 0 to clear the reset condition.
	3	RW	reset_rx_digital	Writing a 1 causes the internal RX digital reset signal to be asserted. You must write a 0 to clear the reset condition.
0x461	[31:0]	RW	phy_serial_loopback	Writing a 1 puts the channel in serial loopback mode.
0x464	[31:0]	RW	pma_rx_set_locktodata	When set, programs the RX CDR PLL to lock to the incoming data.
0x465	[31:0]	RW	pma_rx_set_locktoref	When set, programs the RX clock data recovery (CDR) PLL to lock to the reference clock.

Address	Bit	R/W	Name	Description
0x466	[31:0]	RO	pma_rx_is_lockedtodata	When asserted, indicates that the RX CDR PLL is locked to the RX data, and that the RX CDR has changed from LTR to LTD mode.
0x467	[31:0]	RO	pma_rx_is_lockedtoref	When asserted, indicates that the RX CDR PLL is locked to the reference clock.
0x4A8	0	RW	tx_invpolarity	When set to 1, the TX interface inverts the polarity of the TX data. Inverted TX data is input to the 8B/10B encoder.
	1	RW	rx_invpolarity	When set to 1, the RX channels inverts the polarity of the received data. Inverted RX data is input to the 8B/10B decoder.
	2	RW	rx_bitreversal_enable	When set to 1, enables bit reversal on the RX interface. The RX data is input to the word aligner.
	3	RW	rx_bytereversal_enable	When set, enables byte reversal on the RX interface. The RX data is input to the byte deserializer.
	4	RW	force_electrical_idle	When set to 1, forces the TX outputs to electrical idle.
0x4A9	0	R	rx_syncstatus	When set to 1, indicates that the word aligner is synchronized to incoming data.
	1	R	rx_patterndetect	When set to 1, indicates the 1G word aligner has detected a comma.
	2	R	rx_rlv	When set to 1, indicates a run length violation.
	3	R	rx_rmfifodatainserted	When set to 1, indicates the rate match FIFO inserted code group.
	4	R	rx_rmfifodatadeleted	When set to 1, indicates that rate match FIFO deleted code group.
	5	R	rx_disperr	When set to 1, indicates an RX 8B/10B disparity error.
	6	R	rx_errdetect	When set to 1, indicates an RX 8B/10B error detected.

Creating a 1G/10GbE Design

Here are the steps you must take to create a 1G/10GbE design using the 1G/10GbE PHY IP.

1. Generate the 1G/10GbE PHY with the required parameterization.

   Unlike in the 1G/10GbE PHY IP Core for Stratix V devices, for Arria 10 devices, the Reconfiguration Block is included 1G/10bE PHY. This Reconfiguration Block provides the Avalon-MM interface that you can use to read and write to PHY registers. All read and write operations must be adhere to Avalon specification.

2. Instantiate a reset controller using the Transceiver Reset Controller Megafunction in the MegaWizard Plug-In Manager. Connect the power and reset signals between the 1G/10GbE PHY and the reset controller.

3. Instantiate one TX PLL for the 1G data rate and one TX PLL for the 10G data rate. Connect the high speed serial clock and PLL lock signals between 1G/10GbE PHY and TX PLLs. You can use any combination of fPLLs, ATX, or CMU PLLs.
4. Use the `tx_pma_divclk` from 1G/10GbE PHY or generate a fPLL to create the 156.25 MHz XGMII clock from the 10G reference clock.

   Unlike in the 1G/10GbE PHY IP Core for Stratix V devices, no Memory Initialization Files (**.mif**) are required for the 1G/10GbE design in Arria 10 devices.
5. Complete the design by creating a top level module to connect all the IP (1G/10GbE PHY IP, PLL IP and Reset Controller) blocks.

Related Information

- **fPLL** on page 3-11
- **CMU PLL** on page 3-19
- **ATX PLL** on page 3-3
- **Using the Altera Transceiver PHY Reset Controller** on page 4-8
- **1G/10GbE PHY Functional Description** on page 2-152

Design Guidelines

Consider the following guidelines while designing with 1G/10GbE PHY.

Using the 1G/10GbE PHY without the Sequencer

The sequencer brings up channel-based initial datapath and performs parallel detection. To use the 1G/10GbE PHY without the sequencer, turn off the **Enable automatic speed detection** parameter.

Turning off the sequencer results in the following additional ports:

- `rc_busy`
- `start_pcs_reconfig`
- `mode_1g_10gbar`

These ports perform manual reconfiguration. The following figure shows how these ports are used for 1G and 10G configuration.

Figure 2-64: Timing for Reconfiguration without the Sequencer

Channel Placement Guidelines

The channels of multi-channel 1G/10G designs do not need to be placed contiguously. However, channels instantiated in different transceiver banks require PLLs in the same bank. Currently the Quartus II software version 13.1 for Arria 10 devices does not support PMA bonding for 10GBASE-KR PHY IP.

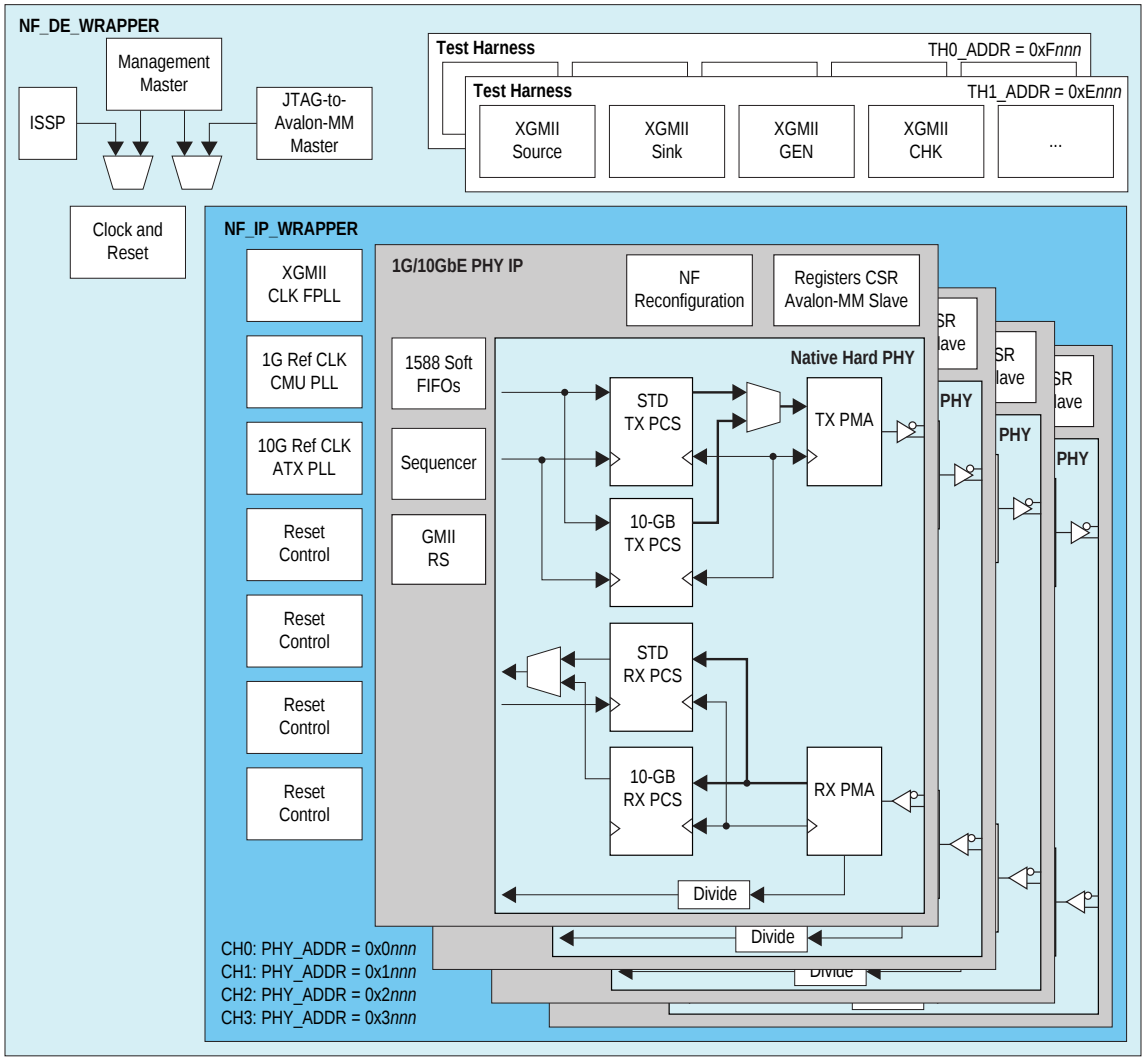
Design Example

Altera provides a design example to assist you in integrating your Ethernet PHY IP into your complete design.

The MAC and PHY design example instantiates the 1G/10GbE PHY IP along with the 1G/10G Ethernet MAC and supporting logic. It is part of the Quartus II 13.1 installation and is located in the *<quartus2_install_dir>/ip* subdirectory. For more information about this example design, refer to the **Ethernet MAC MegaCore Function User Guide**

A design example that includes on the PHY instantiates the 1G/10G PHY and its supporting logic. This design example is available on the Altera wiki. The following figure shows the block diagram of the 1G/10GbE PHY only design example. The default configuration includes two channels for backplane Ethernet and two channels for line-side(1G/10G) applications.

Figure 2-65: 1G/10GbE PHY Only Design Example



Related Information

Arria 10 Transceiver PHY Design Examples

Simulation Support

The 1G/10GbE and 10GBASE-KR PHY IP Core supports ModelSim Verilog and ModelSim VHDL, VCS Verilog, and VCS VHDL simulation. Arria 10 devices also support NCSIM Verilog and NCSIM VHDL simulation. The MegaWizard Plug-In Manager generates an IP functional simulation model when you press the **Finish** button.

TimeQuest Timing Constraints

To pass timing analysis, you must decouple the clocks in different time domains. The necessary Synopsys Design Constraints File (**.sdc**) timing constraints for the are included in the top-level wrapper file.

Acronyms

This table defines some commonly used Ethernet acronyms.

Table 2-122: Ethernet Acronyms

Acronym	Definition
AN	Auto-Negotiation in Ethernet as described in Clause 73 or of IEEE 802.3ap-2007.
BER	Bit Error Rate.
DME	Differential Manchester Encoding.
FEC	Forward error correction.
GMII	Gigabit Media Independent Interface.
KR	Short hand notation for Backplane Ethernet with 64b/66b encoding.
LD	Local Device.
LT	Link training in backplane Ethernet Clause 72 for 10GBASE-KR and 40GBASE-KR4.
LP	Link partner, to which the LD is connected.
MAC	Media Access Control.
MII	Media independent interface.
OSI	Open System Interconnection.
PCS	Physical Coding Sublayer.
PHY	Physical Layer in OSI 7-layer architecture, also in Altera device scope is: PCS + PMA.
PMA	Physical Medium Attachment.
PMD	Physical Medium Dependent.
SGMII	Serial Gigabit Media Independent Interface.
WAN	Wide Area Network.
XAUI	10 Gigabit Attachment Unit Interface.

PCI Express

Arria 10 transceivers can be used to implement a complete PCIe solution for Gen1, Gen2, and Gen3, at data rates of 2.5, 5.0, and 8 Gbps.

You can configure the transceivers for PCIe functionality using one of the following methods:

- **Arria 10 Hard IP for PCI Express**

This is a complete PCI Express solution that includes the Transaction, Data Link, and PHY MAC layers. The Hard IP solution contains dedicated hard logic, which is connected to the transceiver PHY interface.

Note: For more information, refer to the *Arria 10 Hard IP for PCI Express User Guide*.

- **Native PHY IP in PIPE Gen1/Gen2/Gen3 Transceiver Configuration Rules**

Native PHY can be used to configure the transceiver in PCIe mode, giving access to the PIPE interface (commonly called PIPE mode in transceivers). This mode enables you to connect the transceiver to a third-party MAC to create a complete PCIe solution.

This section will focus on the implementation and configuration details for Native PHY IP in PIPE Gen1/Gen2/Gen3 transceiver configuration rules.

The PIPE specification (version 3.0) provides implementation details for a PCIe-compliant physical layer. The Native PHY IP for PIPE Gen1, Gen2, and Gen3 supports ×1, ×2, ×4, or ×8 operation for a total aggregate bandwidth ranging from 2 to 64 Gbps. In a x1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. The ×2, ×4, and ×8 configurations support channel bonding for two-lane, four-lane, and eight-lane links. In these bonded channel configurations, the PCS and PMA blocks of all bonded channels share common clock and reset signals.

Gen1 and Gen2 modes use 8B/10B encoding, which has a 20% overhead to overall link bandwidth. Gen3 modes use 128b/130b encoding, which has an overhead of less than 2%. Gen1 and Gen2 modes use the Standard PCS, while Gen3 mode uses the Gen3 PCS for its operation.

Table 2-123: Transceiver Solutions

Support	Arria 10 Hard IP for PCI Express	Native PHY IP for PCI Express (PIPE)
Gen1, Gen2, and Gen3 data rates	Yes	Yes
MAC, data link, and transaction layer	Yes	User implementation in FPGA core
Transceiver interface	Hard IP through PIPE 3.0 based interface	• PIPE 2.0 for Gen1 and Gen2 • PIPE 3.0 based for Gen3 with Gen1/ Gen2 support

Related Information

- **Arria 10 Hard IP for PCI Express User Guide**
- **Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 2.0**
- **Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 3.0**

Transceiver Channel Datapath for PIPE

Figure 2-66: Transceiver Channel Datapath for PCIe Gen1/Gen2 Configuration with Gen3 Disabled

The following figure shows the Arria 10 transmitter and receiver channel datapath for PIPE Gen1/Gen2 configurations when using PIPE configuration with Gen3 disabled. In this configuration, the transceiver connects to a PIPE interface.

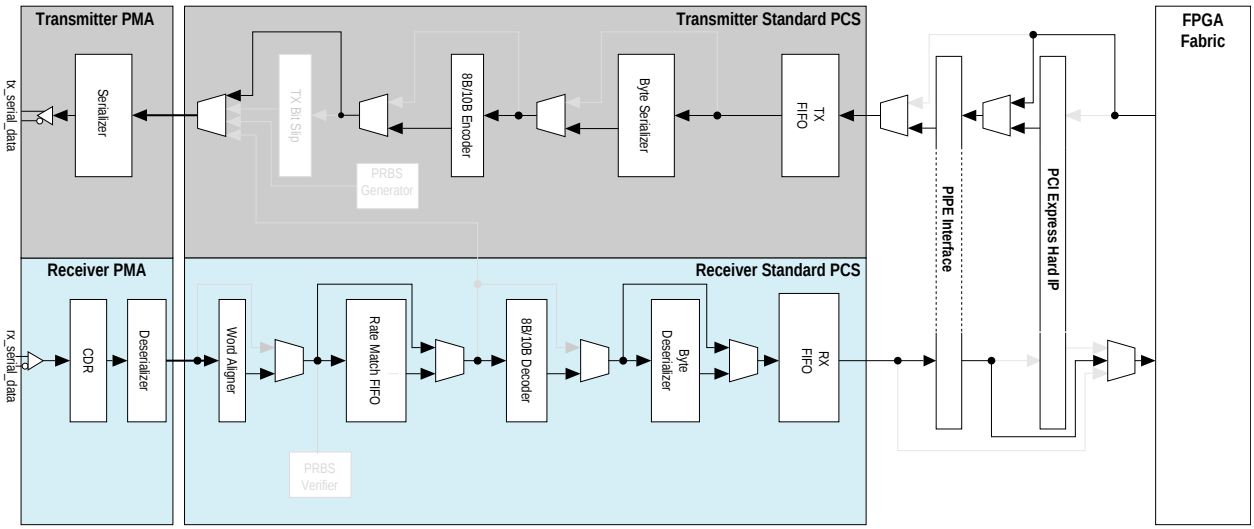
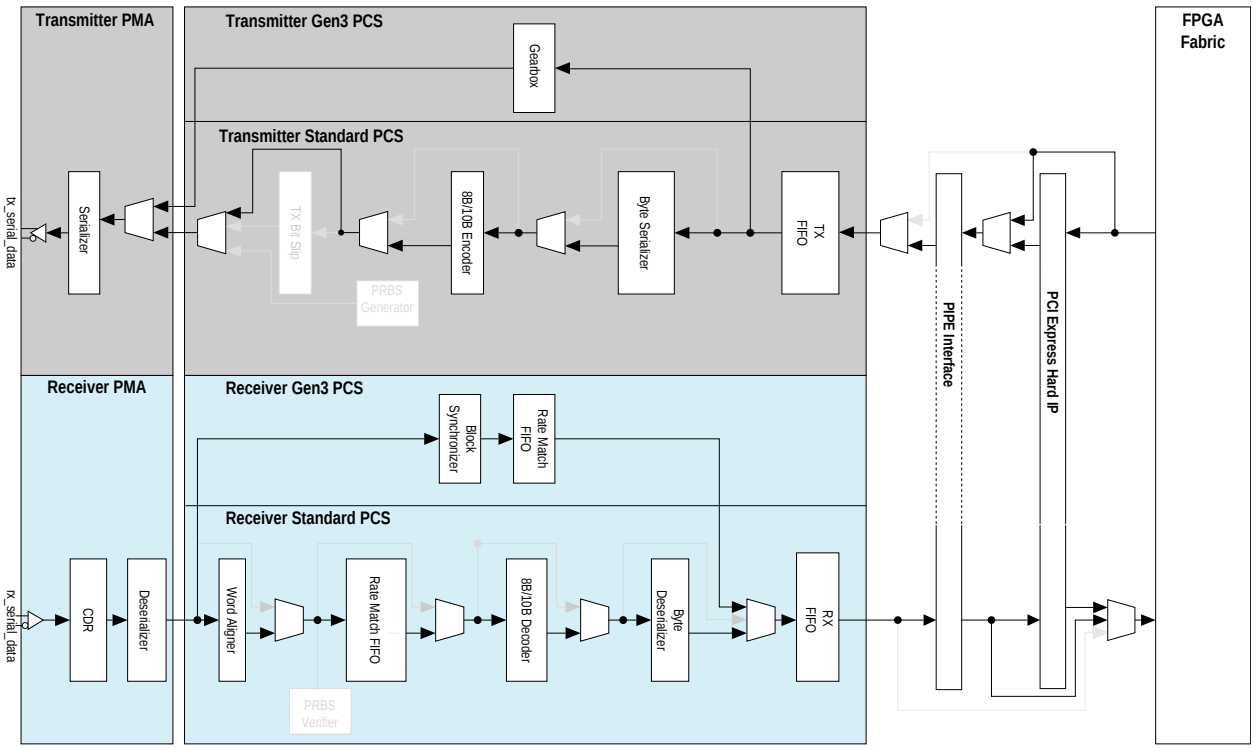


Figure 2-67: Transceiver Channel Datapath for PCIe Gen1/Gen2/Gen3 Configurations

The following figure shows the Arria 10 transmitter and receiver channel datapath for PCIe Gen1/Gen2/Gen3 configurations with a 32-bit PIPE 3.0 based interface.



Supported PIPE Features

The features supported for a PCIe configuration differ according to Gen1, Gen2, or Gen3 configurations.

Table 2-124: Supported Features for PCIe Configurations

Protocol Feature	Gen1 (2.5 Gbps)	Gen2 (5 Gbps)	Gen3 (8 Gbps)
x1, x2, x4, x8 link configurations	Yes	Yes	Yes
PCIe-compliant synchronization state machine	Yes	Yes	Yes
±300 ppm (total 600 ppm) clock rate compensation	Yes	Yes	Yes
Transmitter driver electrical idle	Yes	Yes	Yes
Receiver Detection	Yes	Yes	Yes
8B/10B encoder/decoder disparity control	Yes	Yes	N/A
128B/130B encoder/decoder	N/A	N/A	Yes (supported through the Gearbox)
Scrambler/Descrambler	N/A	N/A	Yes (implemented in FPGA fabric)
Power state management	Yes	Yes	Yes
Receiver PIPE status encoding (pipe_rxstatus[2:0])	Yes	Yes	Yes
Dynamic switching between 2.5 Gbps and 5 Gbps signaling rate	N/A	Yes	N/A
Dynamic switching between 2.5 Gbps, 5 Gbps, and 8 Gbps signaling rate	N/A	N/A	Yes
Dynamic transmitter margining for differential output voltage control	N/A	Yes	Yes
Dynamic transmitter buffer de-emphasis of -3.5 dB and -6 dB	N/A	Yes	Yes
Dynamic Gen3 transceiver pre-emphasis, de-emphasis, and equalization	N/A	N/A	Yes
PCS PMA interface width	Gen1 = 10	Gen2 = 10	Gen3 = 32

Related Information

- **Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 2.0**
- **Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 3.0**
- **Arria 10 Standard PCS Architecture** on page 5-31
  For more information about PIPE Gen1 and Gen2.
- **PCIe Gen3 PCS Architecture**
  For more information about PIPE Gen3.

Gen1/Gen2 Features

In a PIPE configuration, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE configuration is based on PIPE 2.0 specification. If you use a PIPE configuration, you must implement the PHY-MAC layer using soft IP in the FPGA fabric.

Dynamic Switching Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps)

In a PIPE configuration, Native PHY IP provides an input signal (pipe_rate) that is functionally equivalent to the RATE signal specified in the PCIe specification. A low-to-high transition on this input signal (pipe_rate) initiates a data rate switch from Gen1 to Gen2. A high-to-low transition on the input signal initiates a data rate switch from Gen2 to Gen1.

Transmitter Electrical Idle Generation

The PIPE interface block in Arria 10 devices puts the transmitter buffer in the channel in an electrical idle state when the electrical idle input signal is asserted. During electrical idle, the transmitter buffer differential and common configuration output voltage levels are compliant to the PCIe Base Specification 2.0 for both PCIe Gen1 and Gen2 data rates.

The PCIe specification requires the transmitter driver to be in electrical idle in certain power states.

Note: For more information about input signal levels required in different power states, refer to "Power State Management" in the next section.

Power State Management

The PCIe specification defines four power states—P0, P0s, P1, and P2—that the physical layer device must support to minimize power consumption:

- P0 is the normal operating state during which packet data is transferred on the PCIe link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PIPE interface in Arria 10 transceivers provides an input port for each transceiver channel configured in a PIPE configuration.

Note: When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires the physical layer device to implement power saving measures. Arria 10 transceivers do not implement these power saving measures except for putting the transmitter buffer in electrical idle in the lower power states.

8B/10B Encoder Usage for Compliance Pattern Transmission Support

The PCIe transmitter transmits a compliance pattern when the Link Training and Status State Machine (LTSSM) enters the Polling.Compliance substate. The Polling.Compliance substate is used to assess if the transmitter is electrically compliant with the PCIe voltage and timing specifications.

Receiver Electrical Idle Inference (IEI)

The PCIe protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition with analog circuitry.

In all PIPE configurations, (x1, x2, x4, and x8), each receiver channel PCS has an optional Electrical Idle Inference module that implements the electrical idle inference conditions specified in the PCIe Base Specification 2.0.

Altera implements IEI for Gen1 and Gen2 modes. For Gen1 mode, IEI can be done by signal detect or the inference mechanism. For Gen2 mode, IEI is done by inference mechanism. In Gen2 mode, do not use signal detect.

Table 2-125: Electrical Idle Inference Conditions in Gen1/Gen2

LTSSM State	eidleinfersel[2:0]	Gen1	Gen2
L0	3'b100	Absence of Skip Ordered Set in 128 µs window	Absence of Skip Ordered Set in 128 µs window
Recovery.RcvrCfg	3'b101	Absence of TS1 or TS2 Ordered Set in 1280 UI interval	Absence of TS1 or TS2 Ordered Set in 1280 UI interval
Recovery.Speed when successful speed negotiation = 1'b1	3'b101	Absence of TS1 or TS2 Ordered Set in 1280 UI interval	Absence of TS1 or TS2 Ordered Set in 1280 UI interval
Recovery.Speed when successful speed negotiation = 1'b0	3'b110	Absence of an exit from Electrical Idle in 2000 UI interval	Absence of an exit from Electrical Idle in 16000 UI interval
Loopback.Active (as slave)	3'b111	Absence of Skip Ordered Set in 128 µs window	N/A

Receiver Status

The PCIe specification requires the PHY to encode the receiver status on a 3-bit status signal (`pipe_rx_status[2:0]`). This status signal is used by the PHY-MAC layer for its operation. The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks, and encodes the status on the `pipe_rx_status[2:0]` signal to the FPGA fabric. The encoding of the status signals on the `pipe_rx_status[2:0]` signal conforms to the PCIe specification.

Receiver Detection

The PIPE interface block in Arria 10 transceivers provides an input signal (`pipe_tx_detectrx_loopback`) for the receiver detect operation required by the PCIe protocol during the Detect state of the LTSSM. When the `pipe_tx_detectrx_loopback` signal is asserted in the P1 power state, the PCIe interface block sends a command signal to the transmitter driver in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state. After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver (that complies with the PCIe input impedance requirements) is present at the far end, the time constant of the step voltage on the trace is higher when compared with the time constant of the step voltage when the receiver is not present. The receiver detect circuitry monitors the time constant of the step signal seen on the trace to determine if a receiver was detected.

Note: For the receiver detect circuitry to function reliably, the transceiver on-chip termination must be used and the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant with the PCIe Base Specification 2.0.

The PIPE core provides a 1-bit PHY status (`pipe_phy_status`) and a 3-bit receiver status signal (`pipe_rx_status[2:0]`) to indicate whether a receiver was detected or not, as per the PIPE 2.0 specifications.

Gen1 and Gen2 Clock Compensation

In compliance with the PIPE specification, Arria 10 receiver channels have a rate match FIFO to compensate for small clock frequency differences up to ±300 ppm between the upstream transmitter and the local receiver clocks.

Things to remember for PIPE clock compensation spec are as follows:

- Insert/delete **one SKP symbol** in a SKP ordered set
- For deletion, no min limit imposed on the number of SKP symbols present in SKP ordered set after deletion. An ordered set may have a bare COM case after deletion.
- For insertion, no max limit imposed on the number of the SKP symbols present in the SKP ordered set after insertion. An ordered set may have more than 5 symbols appear in SKP ordered set after insertion.
- For INSERT/DELETE cases: Flag status will appear on the COM symbol of the SKP ordered set where insertion/deletion occurred.
- For FULL/EMPTY cases: Flag status will appear where character is inserted or deleted.

  Note: When the PIPE interface is on, it will translate the value of the flags to the appropriate `pipe_rx_status` signal.

- The PIPE mode also has a "0 ppm" configuration option that can be used in synchronous systems. The Rate Match FIFO Block is not expected to do any clock compensation in this configuration, but latency will be minimized.

Figure 2-68: Rate Match Deletion

The figure below shows an example of rate match deletion in the case where two /K28.0/ SKP symbols must be deleted. Only one /K28.0/ SKP symbol is deleted per SKP ordered set received.

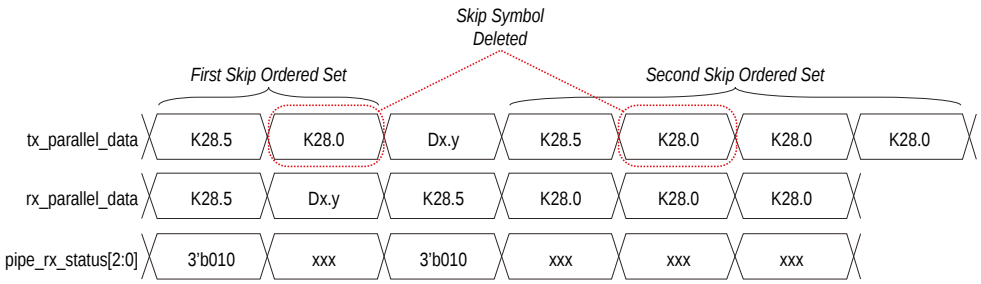


Figure 2-69: Rate Match Insertion

The figure below shows an example of rate match insertion in the case where two SKP symbols must be inserted. Only one /K28.0/ SKP symbol is inserted per SKP ordered set received.

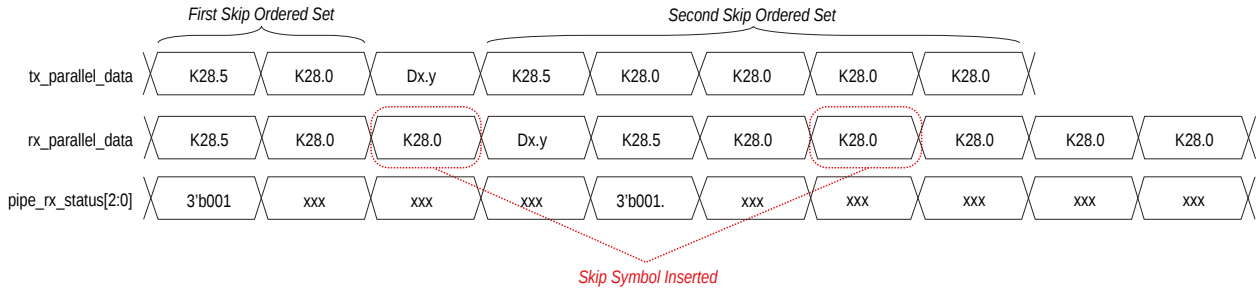


Figure 2-70: Rate Match FIFO Full

The rate match FIFO in PIPE mode automatically deletes the data byte that causes the FIFO to go full and drives pipestatus[2:0] = 3'b101 synchronous to the subsequent data byte. The figure below shows the rate match FIFO full condition in PIPE mode. The rate match FIFO becomes full after receiving data byte D4.

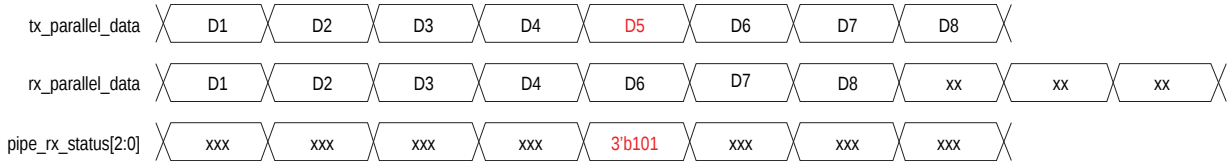
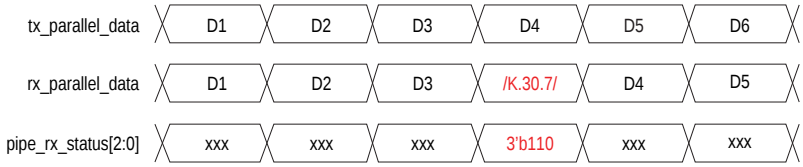


Figure 2-71: Rate Match FIFO Empty

The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to go empty and drives PIPE status[2:0] = 3'b110flag synchronous to the inserted /K30.7/ (9'h1FE). The figure below shows rate match FIFO empty condition in PIPE mode. The rate match FIFO becomes empty after reading out data byte D3.



PIPE 0 ppm

The PIPE mode also has a "0 ppm" configuration option that can be used in synchronous systems. The Rate Match FIFO Block is not expected to do any clock compensation in this configuration, but latency will be minimized (to 3 or 4 cycles).

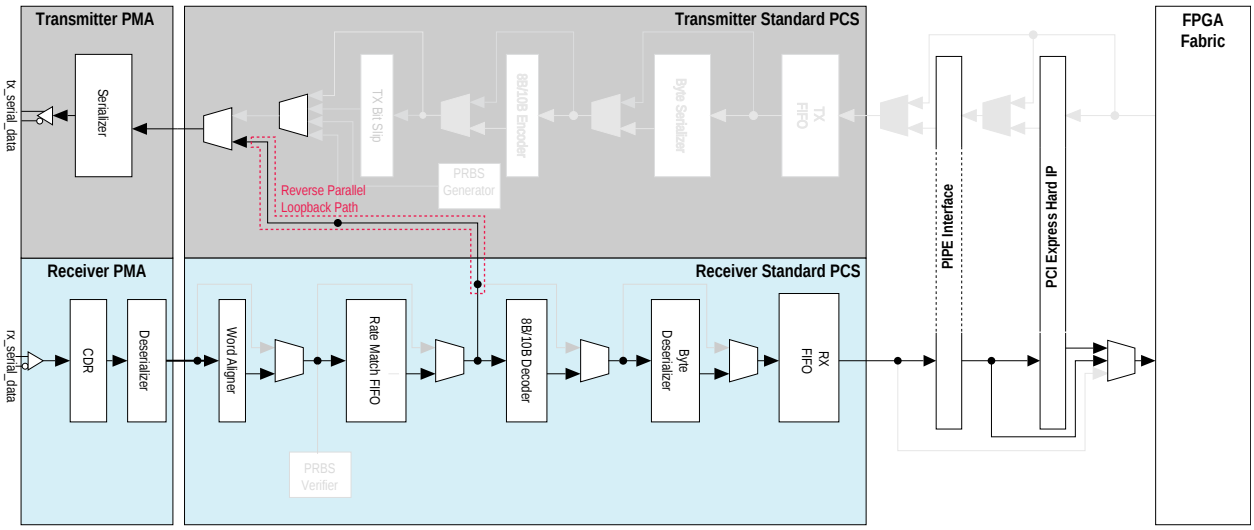
PCIe Reverse Parallel Loopback

PCIe reverse parallel loopback is only available in a PCIe functional configuration for Gen1, Gen2, and Gen3 data rates. The received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer. The data is then looped back to the transmitter serializer and transmitted out through the transmitter buffer. The received data is also available to the FPGA fabric through the

rx_parallel_data port. This loopback mode is based on PCIe specification 2.0. Arria 10 devices provide an input signal to enable this loopback mode.

Note: This is the only loopback option supported in PIPE configurations.

Figure 2-72: PCIe Reverse Parallel Loopback Mode Datapath



Related Information

Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 2.0

Arria 10 Standard PCS Architecture on page 5-31

Gen3 Features

The following sub-section explains how the various PIPE features are supported by the Arria 10 transceiver block.

The PCS supports PIPE 3.0 base specification. The PIPE interface has been expanded to a 32-bit wide PIPE 3.0-based interface. The PIPE interface controls PHY functions such as transmission of electrical idle, receiver detection, and speed negotiation and control.

Auto-Speed Negotiation

PIPE Gen3 mode enables ASN between Gen1 (2.5 Gbps), Gen2 (5.0 Gbps), and Gen3 (8.0 Gbps) signaling data rates. The signaling rate switch is accomplished through frequency scaling and configuration of the PMA and PCS blocks using a fixed 32-bit wide PIPE 3.0-based Interface.

The PMA switches clocks between Gen1, Gen2, and Gen3 data rates. For a non-bonded x1 channel, an ASN module facilitates speed negotiation in that channel. For bonded x2, x4, and x8 channels, the ASN module selects the master channel to control the rate switch. The master channel distributes the speed change request to the other PMA and PCS channels.

The PCIe Gen3 speed negotiation process is initiated by a rate change requested from Hard IP or FPGA FABRIC. The ASN then places the PCS in reset, and dynamically shuts down the clock paths to disengage the current active state PCS (either Standard PCS or Gen3 PCS). If a switch to or from Gen3 is requested,

the ASN automatically selects the correct PCS clock paths and datapath selection in the multiplexers. The ASN block then sends a request to the PMA block to switch the data rate and waits for a rate change done signal for confirmation. When the PMA completes the rate change and sends confirmation to the ASN block, the ASN enables the clock paths to engage the new PCS block and releases the PCS reset. Successful completion of this process is indicated by assertion of the `pipe_phy_status` signal by the ASN block.

Note: In Native PHY IP PIPE configuration, you must set `pipe_rate[1:0]` to initiate the transceiver datarate switch sequence.

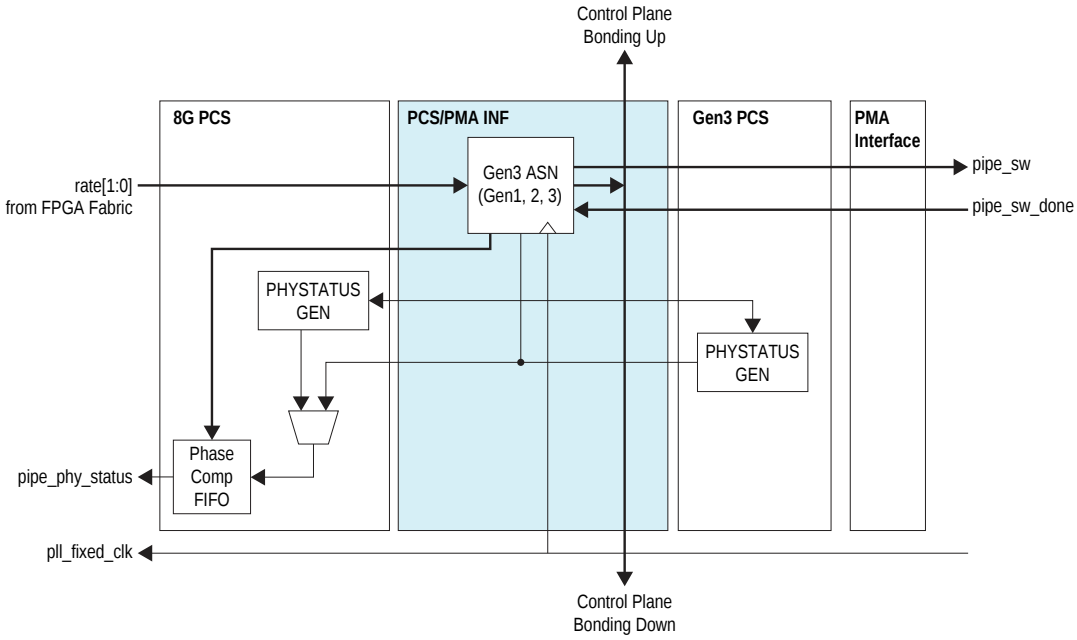
Rate Switch

This section provides an overview of auto rate change between PIPE Gen1 (2.5 Gbps), Gen2 (2.5 Gbps), and Gen3 (8.0 Gbps) mode.

The switches among Gen1, Gen2, and Gen3 rates involve reconfiguration of PMA and PCS settings. PMA needs to re-lock and provide a TX PLL clock, and its CDR will also lock at a new incoming data rate. PIPE interface clock rate is also adjusted to match the data throughput. In Arria 10 devices, there is only one common ASN block located in the PMA PCS interface handling all PIPE speed change.

Figure 2-73: Rate Switch Change

The block-level diagram below shows a high level connectivity between ASN and 8G PCS and Gen3 PCS.

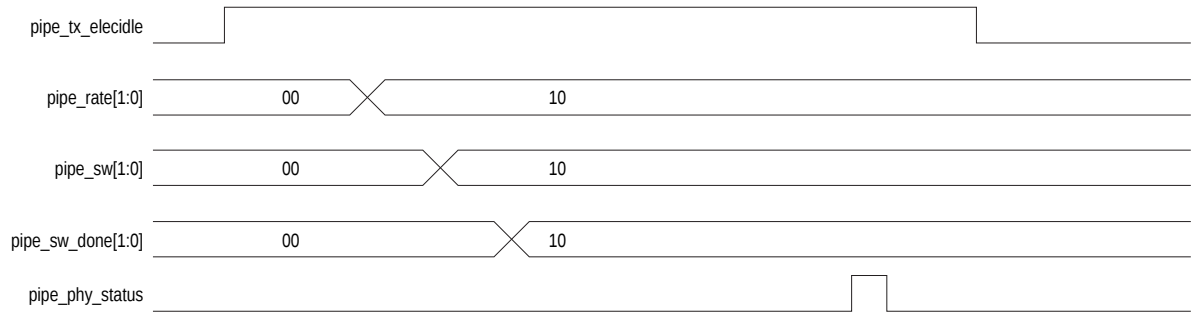


The sequence of speed change between Gen1, Gen2, and Gen3 occurs as follows:

1. The PHY MAC layer implemented in FPGA Fabric requests a rate change through `pipe_rate[1:0]`.
2. The ASN block waits for Phase compensation FIFO to flush out data. Then ASN block asserts the PCS reset.
3. The ASN asserts clock shutdown signal to 8G PCS and Gen3 PCS to dynamically shutdown the clock.
4. The ASN asserts the clock and data multiplexer selection signals. This is performed only when rate changes from, or to Gen3 speed.
5. The ASN sends rate change request to PMA. This is done through `pipe_sw[1:0]` output signal.

6. The ASN waits for rate change done from PMA. This is done through continuously monitoring the `pipe_sw_done[1:0]` input signal.
7. After the ASN receives `pipe_sw_done`, it deasserts the clock shut down signals to release the clock.
8. The ASN deasserts the PCS reset.
9. The ASN sends the speed change completion to PHY-MAC interface. This is done through the `pipe_phy_status` signal to PHY-MAC interface.

Figure 2-74: Speed Change Sequence



Gen3 Transmitter Electrical IDLE Generation

In the PIPE 3.0-based interface the user may place the transmitter in electrical idle during low power states. Before the transmitter enters electrical idle, the user must send the Electrical Idle ordered set consisting of 16 symbols with value 0x66. During electrical idle, the transmitter differential and common mode voltage levels are based on the *PCIe Base Specification 3.0*.

Gen3 Clock Compensation

This mode can be enabled from the MegaWizard™ GUI when using the Gen3 PIPE transceiver configuration rule.

To accommodate PCIe protocol requirements and to compensate for clock frequency differences of up to ±300 ppm between source and termination equipment, receiver channels have a rate match FIFO. The rate match FIFO adds or deletes four SKP characters (32 bits) to keep the FIFO from becoming empty or full. It monitors the block synchronizer for a `skip_found` signal. If the rate match FIFO is almost full, the FIFO deletes four SKP characters. If the rate match FIFO is nearly empty, the FIFO inserts a SKP character at the start of the next available SKP ordered set. FIFO full, empty, insertion and deletion is indicated by `pipe_rx_status`.

Note: Refer to the Gen1 and Gen2 clock compensation section for waveforms.

Related Information

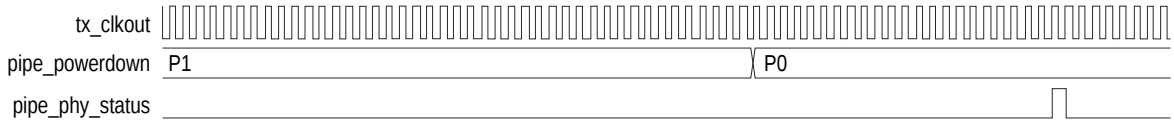
Gen1 and Gen2 Clock Compensation on page 2-185

Gen3 Power State Management

The PCIe base specification defines low power states for PHY layer devices to minimize power consumption. The Gen3 PCS does not implement these power saving measures, except when placing the transmitter driver in electrical idle state in the low power states. In P2 low power state, the transceivers do not disable the PIPE block clock.

Figure 2-75: P1 to P0 Transition

The figure below shows the transition from P1 to P0 with completion provided by `pipe_phy_status`.



CDR Control

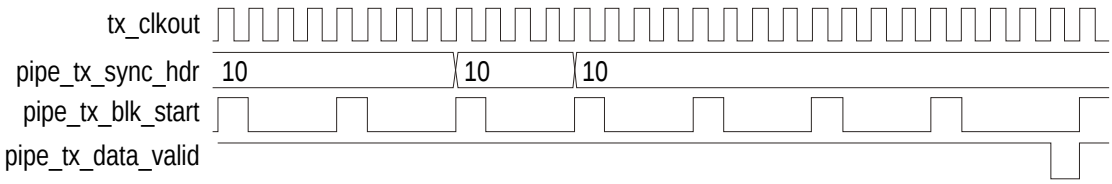
The CDR control block controls the PMA CDR to obtain bit and symbol alignment and deskew within the allocated time, and generates status signals for other PCS blocks. The PCIe base specification requires that the receiver L0s power state exit time be a maximum of 4 ms for Gen1, 2 ms for Gen2, and 4 ms for Gen3 signaling rates. The transceivers have an improved CDR control block to accommodate fast lock times when the CDR must relock to the new multiplier/divider settings when entering or exiting Gen3 speeds.

Gearbox

As per PIPE 3.0 spec, for every 128 bits that are moved across the Gen3 PCS, the PHY must transmit 130 bits of data. Altera uses the `pipe_tx_data_valid` signal every 16 blocks of data to transmit the built-up backlog of 32 bits of data.

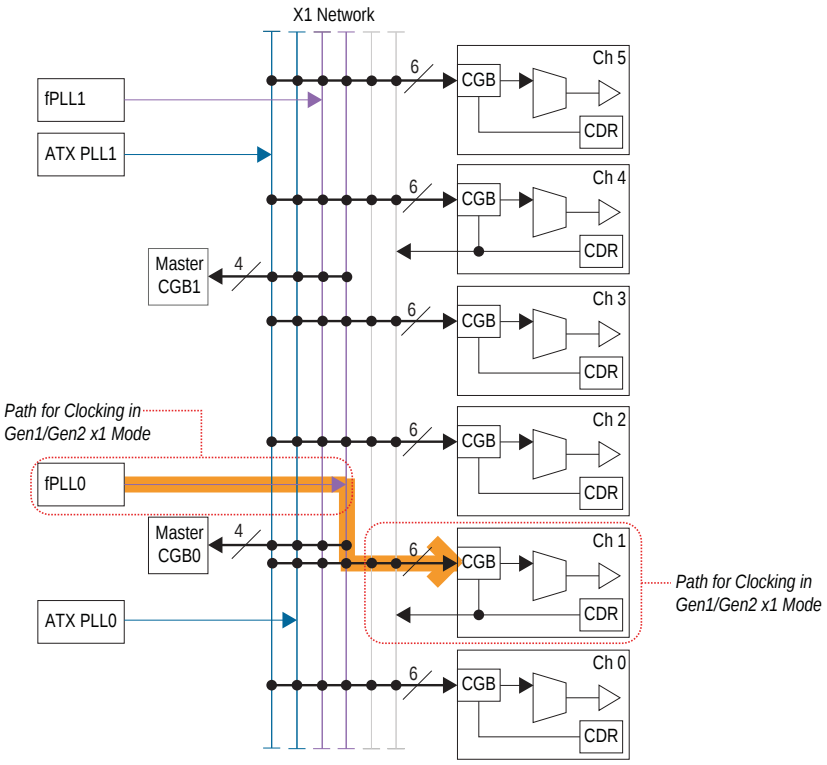
The 130-bit block is received as follows in the 32-bit data path: 34 (32+2bit sync header), 32, 32, 32. During the first cycle, the gearbox converts the 34-bit input data to 32-bit data. During the next 3 clock cycles, the gearbox merges bits from adjacent cycles. In order for the gearbox to work correctly, a gap must be provided in the data for every 16 shifts as each shift is 2 bits for converting the initial 34-bit to 32-bit in the gearbox. After 16 shifts, the gearbox has an extra 32 bits of data that were transmitted out, thus requiring a gap in the input data stream. This is achieved by driving `pipe_tx_data_valid` low for one cycle after every 16 blocks of data.

Figure 2-76: Gen3 Data Transmission



How to Connect TX PLLs for PIPE Gen1, Gen2, and Gen3 Mode

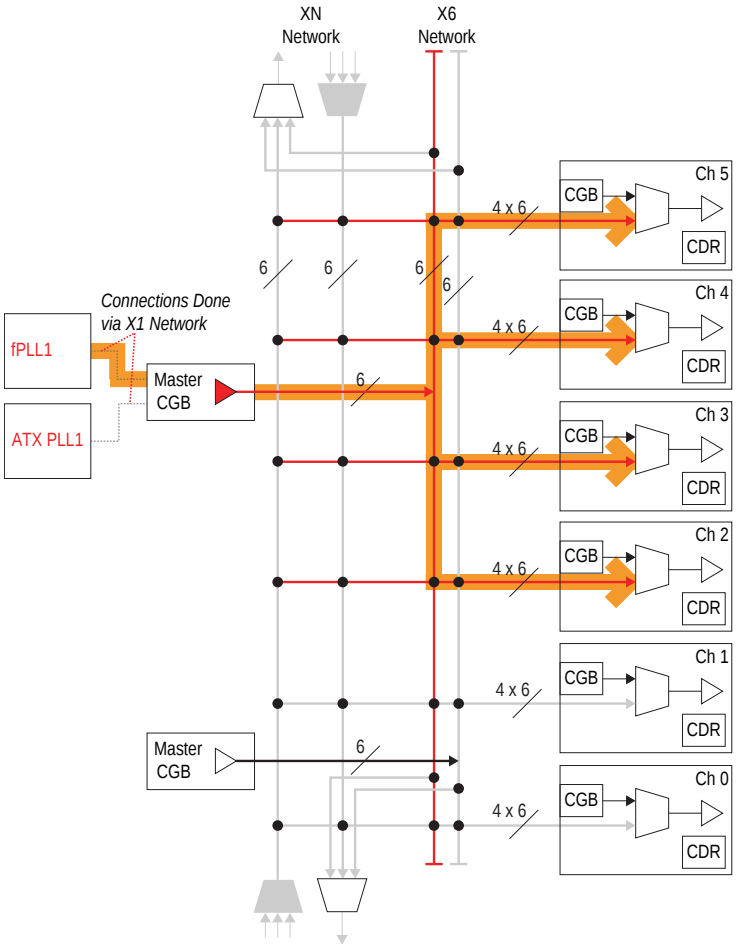
Figure 2-77: Use ATX PLL or fPLL for Gen1/Gen2 x1 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2 x1 mode.
2. Gen1/Gen2 x1 mode uses the ATX PLL or fPLL.
3. Gen1/Gen2 x1 can use any channel from the given bank for which the ATX PLL or fPLL is enabled.
4. Use the pll_pcie_clk from either the ATX PLL or fPLL. This is the hclk required by the PIPE interface.

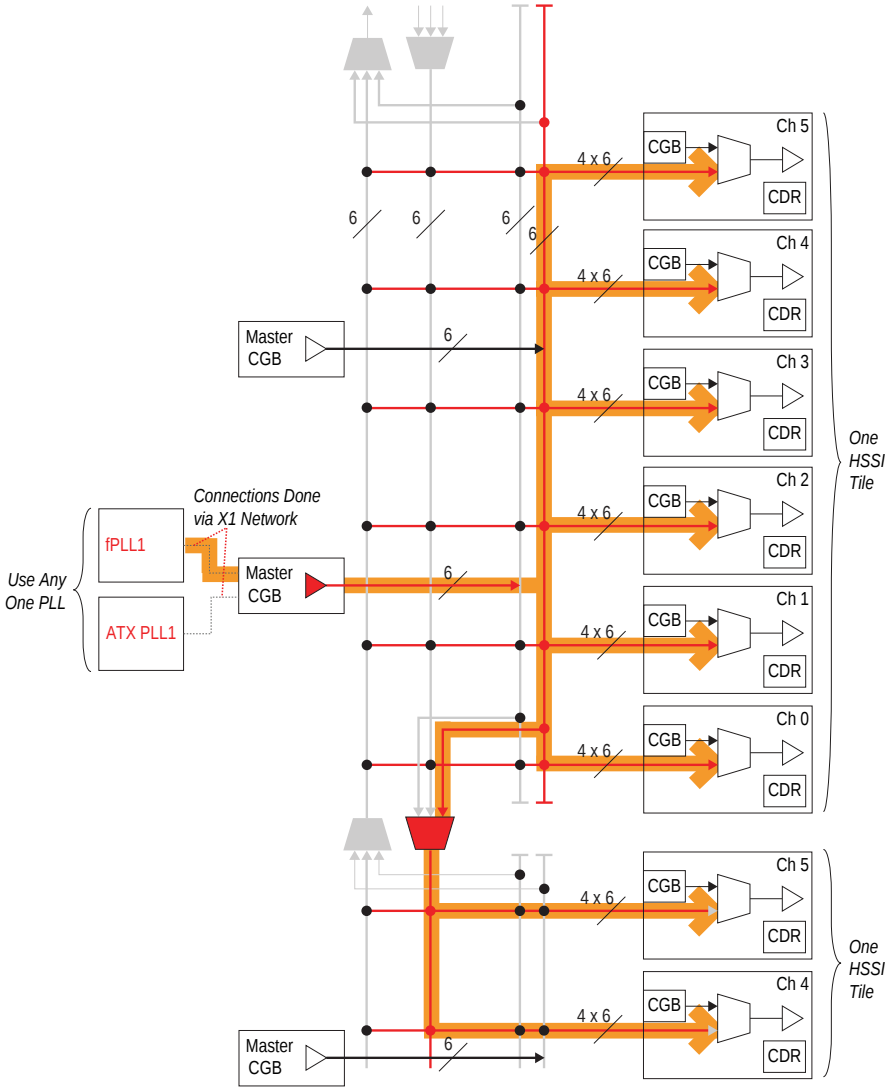
Figure 2-78: Use ATX PLL or fPLL for Gen1/Gen2 x4 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2 x4 mode.
2. The x6 and xN clock networks are used for channel bonding applications.
3. Each master CGB drives one set of x6 clock lines,
4. Gen1/Gen2 x4 mode use the ATX PLL or fPLL.
5. Use the pll_pcie_clk from either the ATX or fPLL. This is the hclk required by the PIPE interface.
6. In this case the Master PCS channel is logical channel 3 (physical channel 4).

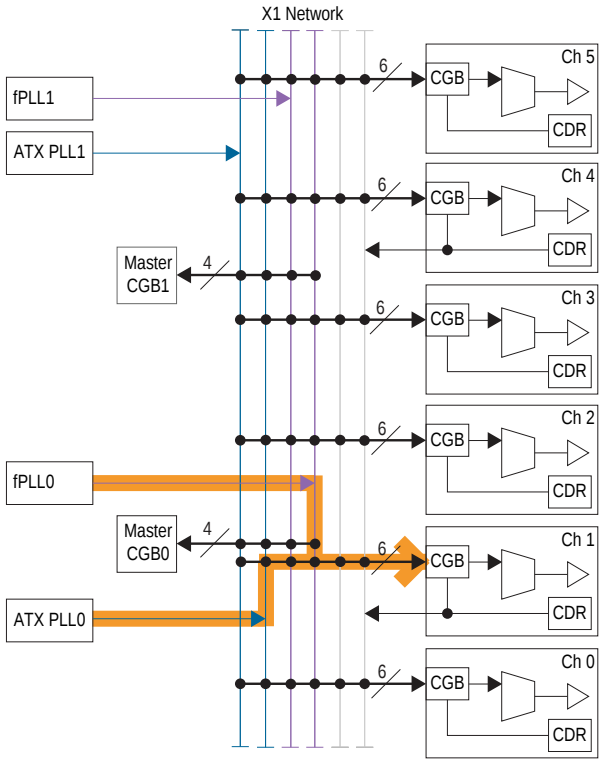
Figure 2-79: Use ATX PLL or fPLL for Gen1/Gen2 x8 Mode



Notes:

1. Figure shown is just one possible combination for the PCIe Gen1/Gen2 x8 mode.
2. The x6 and xN clock networks are used for channel bonding applications.
3. Each master CGB drives one set of x6 clock lines. The x6 lines further drive the xN lines.
4. Gen1/Gen2 x8 mode use the ATX PLL or fPLL.
5. Use the pll_pcie_clk from either the ATX or fPLL. This is the hclk required by the PIPE interface.
6. In this case the Master PCS channel is logical channel 4 (Ch 1 in the top bank).

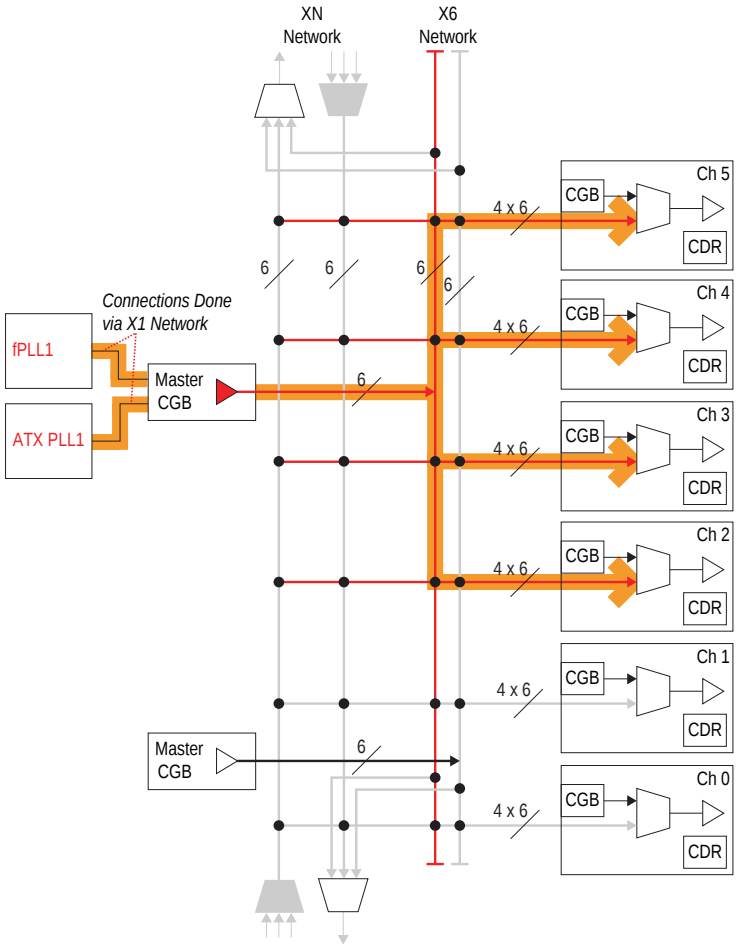
Figure 2-80: Use ATX PLL or fPLL for Gen1/Gen2/Gen3 x1 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2/Gen3 x1 mode.
2. Gen1/Gen2 mode use the fPLL, ONLY.
3. Gen3 mode uses the ATX PLL, ONLY.
4. Use the pll_pcie_clk from the fPLL, configured as Gen1/Gen2. This is the hclk required by the PIPE interface.
5. Select the number of TX PLLs (2) in the Native PHY wizard.

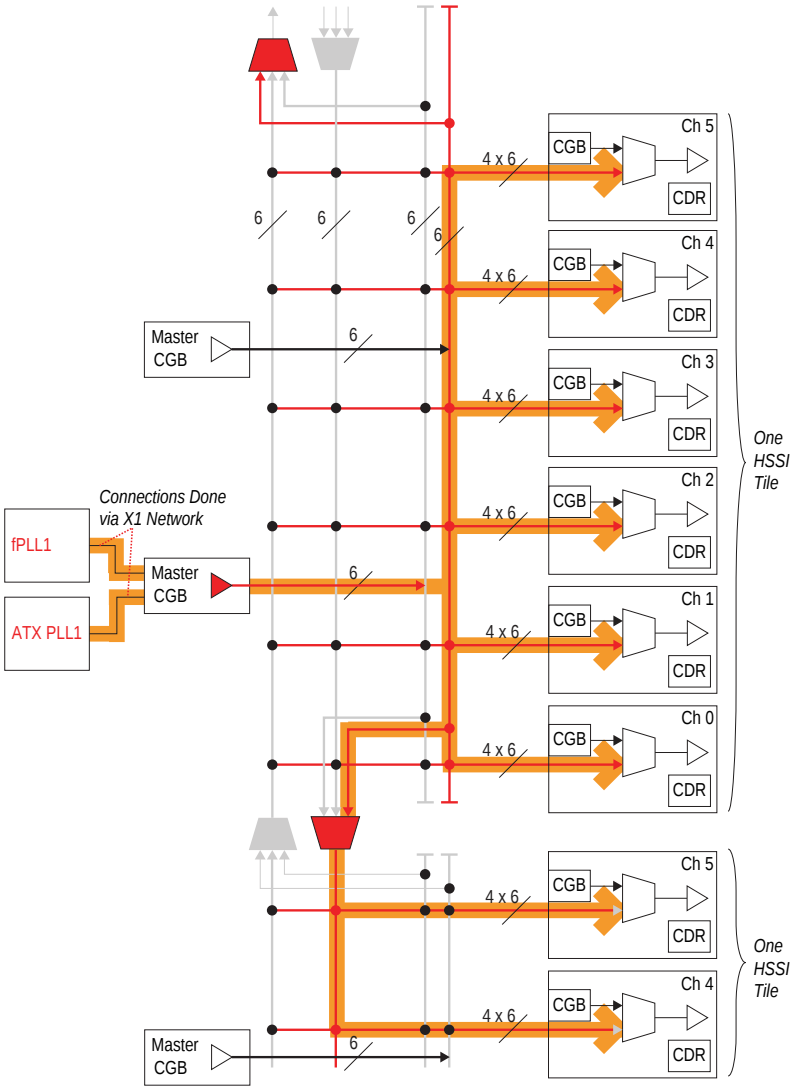
Figure 2-81: Use ATX PLL or fPLL for Gen1/Gen2/Gen3 x4 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2/Gen3 x4 mode.
2. The x6 and xN clock networks are used for channel bonding applications.
3. Each master CGB drives one set of x6 clock lines.
4. Gen1/Gen2 mode use the fPLL, ONLY.
5. Gen3 mode uses the ATX PLL, ONLY.
6. Use the pll_pcie_clk from the fPLL, configured as Gen1/Gen2. This is the hclk required by the PIPE interface.

Figure 2-82: Use ATX PLL or fPLL for Gen1/Gen2/Gen3 x8 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2/Gen3 x8 mode.
2. The x6 and xN clock networks are used for channel bonding applications.
3. Each master CGB drives one set of x6 clock lines. The x6 lines further drive the xN lines.
4. Gen1/Gen2 x8 use the fPLL, ONLY.
5. Gen3 mode uses the ATX PLL, ONLY.
6. Use the pll_pcie_clk from the fPLL, configured as Gen1/Gen2. This is the hclk required by the PIPE interface.

Related Information

- **PIPE Design Example**
  For more information about the PLL MegaWizard configuration for PCIe.
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing clock configurations and configuring PLLs.

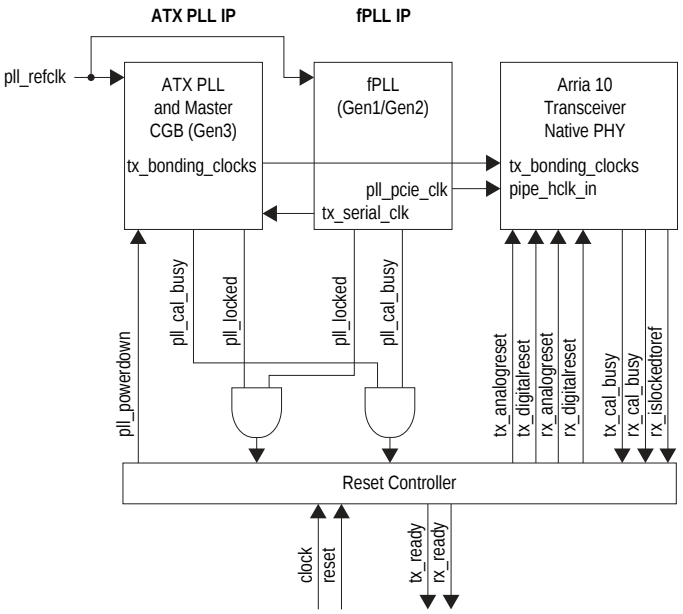
How to Implement PCI Express in Arria 10 Transceivers

Before you begin

You must be familiar with Standard PCS architecture, Gen3 architecture, PLL architecture, and the reset controller before implementing the PCI Express protocol.

1. Open the MegaWizard Plug-In Manager and select the **Arria 10 Transceiver Native PHY IP**. Refer to **Select and Instantiate PHY IP** on page 2-2 for more details.
2. Select **PIPE Gen1/Gen2/Gen3** from the **Transceiver configuration rules** list, located under **Datapath Options**.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for PCI Express** as a starting point. Or, you can use the protocol presets described in ***Transceiver Native PHY Presets*** . You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).
5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Altera Transceiver PHY Reset Controller IP.
7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in **Native PHY IP Ports for PCI Express** to connect the ports.

Figure 2-83: Connection Guidelines for a PIPE Design



Note:

1. This is one possible combination to represent the PIPE Gen3 solution, using the Native PHY.

8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Standard PCS Architecture** on page 5-31
- **PLLs** on page 3-3
  For information about PLL architecture and implementation details.

- **Resetting Transceiver Channels** on page 4-1
  For information about the Reset controller and implementation details.
- **Using PLLs and Clock Networks** on page 3-40
- **Design Example** on page 2-215

Native PHY IP Parameter Settings for PCI Express

Table 2-126: Parameterizing Arria 10 Native PHY IP in PIPE Gen1, Gen2, Gen3 Modes

	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Parameter			
Device speed grade	Fastest	Fastest	Fastest
Message level for rule violations	Error, warning	Error, warning	Error, warning
PCS Options			
Transceiver mode	TX / RX Duplex	TX / RX Duplex	TX / RX Duplex
Initial PCS selection	Standard	Standard	Standard
Number of data channels	Gen1 x1: 1 Channel Gen1 x2: 2 Channel Gen1 x4: 4 Channel Gen1 x8: 8 Channel	Gen2 x1: 1 Channel Gen2 x2: 2 Channel Gen2 x4: 4 Channel Gen2 x8: 8 Channel	Gen3 x1: 1 Channel Gen3 x2: 2 Channel Gen3 x4: 4 Channel Gen3 x8: 8 Channel
Data rate	2.5 Gbps	5 Gbps	8 Gbps
Enable reconfig. between Standard and Enhanced PCS	Off	Off	Off
Enable simplified data interface	Optional. If this option is selected as "OFF" refer to the table for signal mapping, provided at end of this table.	Optional. If this option is selected as "OFF" refer to the table for signal mapping, provided at end of this table.	Optional. If this option is selected as "OFF" refer to the table for signal mapping, provided at end of this table.

Table 2-127: Parameterizing Arria 10 Native PHY IP in PIPE Gen1, Gen2, Gen3 Modes - TX PMA

	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
TX Bonding Options			
TX channel bonding mode	Non-bonded (x1) PMA & PCS Bonding	Non-bonded (x1) PMA & PCS Bonding	Non-bonded (x1) PMA & PCS Bonding
PCS TX channel bonding master	Auto	Auto	Auto

	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Actual PCS TX channel bonding master	0, 1	0, 1	0, 1
TX PLL Options			
TX local clock division factor	N/A	N/A	N/A
Number of TX PLLs	1	1	Gen3 x1: 2 All other modes: 1
Main TX PLL logical index	N/A	N/A	N/A
TX PMA Optional Ports			
Enable `tx_pma_clkout` port	Optional	Optional	Optional
Enable `tx_pma_div_clkout` port	Optional	Optional	Optional
`tx_pma_div_clkout` division factor	Optional	Optional	Optional
Enable `tx_pma_clkslip` port	N/A	N/A	N/A
Enable `tx_pma_qpipullup` port (QPI)	Off	Off	Off
Enable `tx_pma_qpipulldn` port (QPI)	Off	Off	Off
Enable `tx_pma_txdetectrx` port (QPI)	Off	Off	Off
Enable `tx_pma_rxfound` port (QPI)	Off	Off	Off

Table 2-128: Parameterizing Arria 10 Native PHY IP in PIPE Gen1, Gen2, Gen3 Modes - RX PMA

	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
RX CDR Options			
Number of CDR reference Clocks	1	1	1
Selected CDR reference clock	0	0	0
Selected CDR reference clock frequency	100 MHz / 125 MHz	100 MHz / 125 MHz	100 MHz / 125 MHz
PPM detector threshold	300	300	300
Decision Feedback Equalization (DFE)			
Decision feedback equalization mode	Off	Off	Off
RX PMA Optional Ports			

	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Enable `rx_pma_clkout` port	Optional	Optional	Optional
Enable `rx_pma_div_clkout` port	Optional	Optional	Optional
`tx_pma_div_clkout` division factor	Optional	Optional	Optional
Enable `rx_pma_clkslip` port	Optional	Optional	Optional
Enable `rx_pma_qpipulldn` port (QPI)	Off	Off	Off
Enable `rx_is_lockedto-data` port	Optional	Optional	Optional
Enable `rx_is_lockedtoref port` port	Optional	Optional	Optional
Enable `rx_set_locktodata` and `rx_set_locktoref` ports	Optional	Optional	Optional
Enable `rx_seriallpbken` port	Optional	Optional	Optional
Enable PRBS Verifier Control and Status ports	Optional	Optional	Optional

Table 2-129: Parameterizing Arria 10 Native PHY IP in PIPE Gen1, Gen2, Gen3 Modes - Standard PCS

Parameter	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Standard PCS configurations			
Standard PCS / PMA interface width	10	10	10
FPGA Fabric / Standard TX PCS interface width	8, 16	16	32
FPGA Fabric / Standard RX PCS interface width	8, 16	16	32
Enable Standard PCS low latency mode	Off	Off	Off
Phase Compensation FIFO			
TX FIFO mode	`low_latency`	`low_latency`	`low_latency`
Enable fast TX FPGA Fabric interface	Off	Off	Off
RX FIFO Mode	`low_latency`	`low_latency`	`low_latency`
Enable `tx_std_pcfifo_full` port	Optional	Optional	Optional

Parameter	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Enable `tx_std_pcfifo_empty` port	Optional	Optional	Optional
Enable `rx_std_pcfifo_full`	Optional	Optional	Optional
Enable `rx_std_pcfifo_empty` port	Optional	Optional	Optional
Byte Serializer and Deserializer			
TX byte serializer mode	Disabled, Serialize x2	Serialize x2	Serialize x4
RX byte deserializer mode	Disabled, Serialize x2	Serialize x2	Serialize x4
8B/10B Encoder and Decoder			
Enable TX 8B/10B encoder	Enabled	Enabled	Enabled
Enable TX 8B/10B disparity control	On	On	On
Enable RX 8B/10B decoder	Enabled	Enabled	Enabled
Rate Match FIFO			
Rate Match FIFO mode	PIPE	PIPE	PIPE
RX rate match insert / delete -ve pattern (hex)	0x0002f17c	0x0002f17c	0x0002f17c
RX rate match insert / delete +ve pattern (hex)	0x000d0e83	0x000d0e83	0x000d0e83
Enable `rx_std_rmfifo_full` port	Optional	Optional	Optional
Enable `rx_std_rmfifo_empty` port	Optional	Optional	Optional
Word Aligner and Bit Slip			
Enable TX bit slip	Off	Off	Off
Enable `tx_std_bitslip-boundarysel` port	Optional	Optional	Optional
RX word aligner mode	Synchronous State Machine	Synchronous State Machine	Synchronous State Machine
RX word aligner pattern length	10	10	10
RX word aligner pattern (hex)	0x0000 00000000017c	0x0000 00000000017c	0x0000 00000000017c
Number of word alignment patterns to achieve sync	3	3	3
Number of invalid data words to lose sync	16	16	16
Number of valid data words to decrement error count	15	15	15

Parameter	Gen1 PIPE	Gen2 PIPE	Gen3 PIPE
Enable `rx_std_wa_patternalign` port	Optional	Optional	Optional
Enable `rx_std_wa_a1a2size` port	Off	Off	Off
Enable `rx_std_bitslip-boundarysel` port	Optional	Optional	Optional
Enable `rx_bitslip` port	Off	Off	Off
Bit Reversal and Polarity Inversion			
Enable TX bit reversal	Off	Off	Off
Enable TX byte reversal	Off	Off	Off
Enable TX polarity inversion	Off	Off	Off
Enable RX bit reversal	Off	Off	Off
Enable RX byte reversal	Off	Off	Off
Enable RX polarity inversion	Off	Off	Off
Enable `tx_polinv` port	Off	Off	Off
Enable `rx_std_bitrev_ena` port	Off	Off	Off
Enable `rx_std_byterev_ena` port	Off	Off	Off
Enable `rx_polinv` port	Off	Off	Off
Enable `tx_std_elecidle` port	Optional	Optional	Optional
Enable `rx_std_signalde-tect` port	Optional	Optional	Optional
PCIe Ports			
Enable PCIe dynamic datarate switch ports	Off	On	On
Enable PCIe `pipe_hclk_in` and `pipe_hclk_out` ports	On	On	On
Enable PCIe Gen3 analog control ports	Off	Off	On
Enable PCIe electrical idle control and status ports	On	On	On
Enable PCIe `pipe_rx_polarity` port	On	On	On
Enable dynamic reconfiguration	Optional	Optional	Optional

Table 2-130: Parameter Settings When the Simplified Interface Is Enabled

For every 128-bit word.

Signal Name	Gen1	Gen2	Gen3
tx_parallel_data	tx_parallel_data[7:0]	tx_parallel_ data[29:22,7:0]	tx_parallel_ data[40:33,29:22,18:11,7:0]
tx_datak	tx_parallel_data[8]	tx_parallel_data[30,8]	tx_parallel_data[41,30,19,8]
pipe_tx_compliance	tx_parallel_data[9]	tx_parallel_data[31,9]	tx_parallel_data[42,31,20,9]
pipe_tx_elecidle	tx_parallel_data[10]	tx_parallel_ data[32,10]	tx_parallel_data[43,32,21,10]
pipe_tx_detectrx_loopback	tx_parallel_data[46]	tx_parallel_data[46]	tx_parallel_data[46]
pipe_powerdown	tx_parallel_ data[48:47]	tx_parallel_ data[48:47]	tx_parallel_data[48:47]
pipe_tx_margin	tx_parallel_ data[51:49]	tx_parallel_ data[51:49]	tx_parallel_data[51:49]
pipe_tx_swing	tx_parallel_data[53]	tx_parallel_data[53]	tx_parallel_data[53]
rx_parallel_data	rx_parallel_data[7:0]	rx_parallel_ data[39:32,7:0]	rx_parallel_ data[55:48,39:32,23:16,7:0]
rx_datak	rx_parallel_data[8]	rx_parallel_data[40,8]	rx_parallel_data[56,40,24,8]
rx_syncstatus	rx_parallel_data[10]	rx_parallel_ data[42,10]	rx_parallel_data[58,42,26,10]
pipe_phy_status	rx_parallel_data[65]	rx_parallel_data[65]	rx_parallel_data[65]
pipe_rx_valid	rx_parallel_data[66]	rx_parallel_data[66]	rx_parallel_data[66]
pipe_rx_status	rx_parallel_ data[69:67]	rx_parallel_ data[69:67]	rx_parallel_data[69:67]
pipe_tx_deemph	N/A	tx_parallel_data[52]	N/A
pipe_tx_sync_hdr	N/A	N/A	tx_parallel_data[55:54]
pipe_tx_blk_start	N/A	N/A	tx_parallel_data[56]
pipe_tx_data_valid	N/A	N/A	tx_parallel_data[60]
pipe_rx_sync_hdr	N/A	N/A	rx_parallel_data[71:70]
pipe_rx_blk_start	N/A	N/A	rx_parallel_data[72]
pipe_rx_data_valid	N/A	N/A	rx_parallel_data[76]

Native PHY IP Ports for PCI Express

Figure 2-84: Signals and Ports of Native PHY IP for PIPE

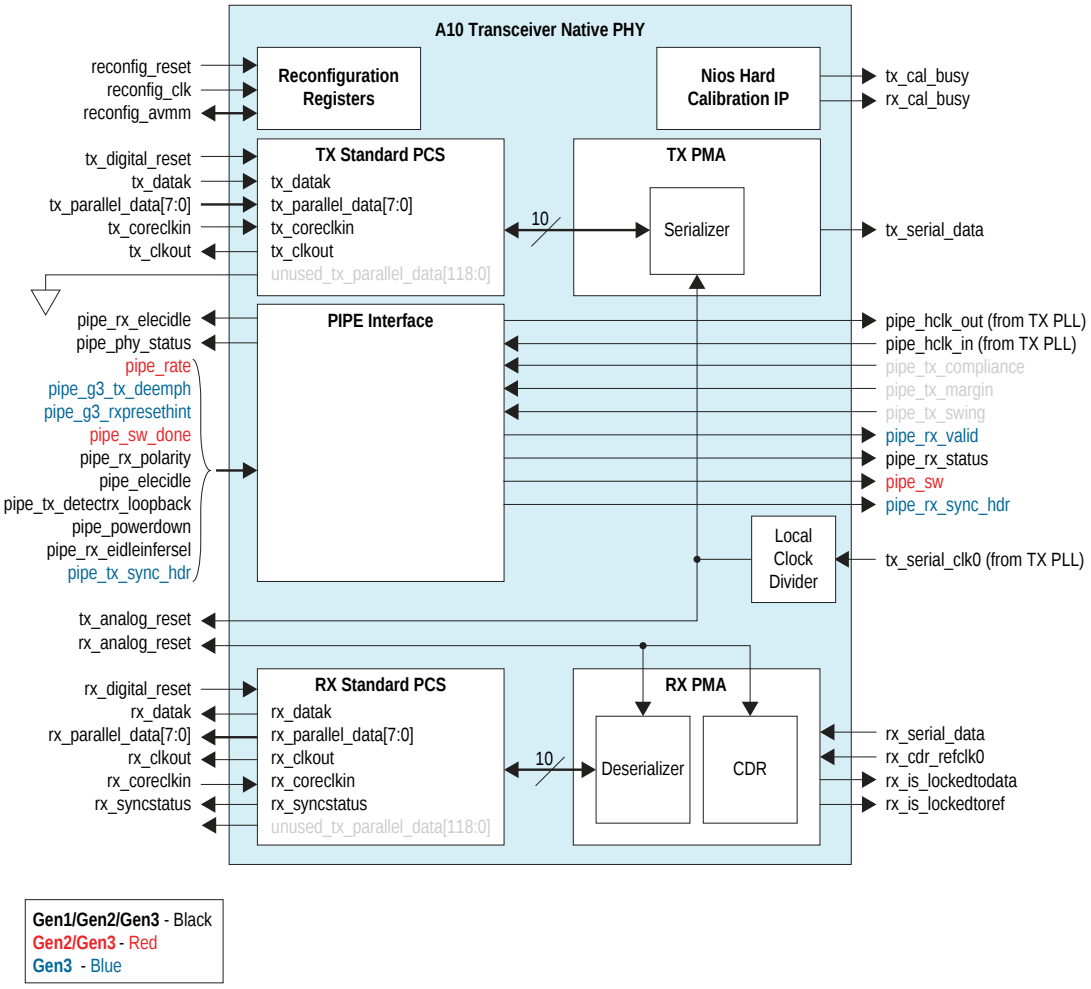


Table 2-131: Ports for Arria 10 Transceiver Native PHY in PIPE Mode

Port	Direction	Clock Domain	Description
Clocks			
rx_cdr_refclk0	In	N/A	The 100/125 MHz input reference clock source for the PHY's TX and RX PLL.
tx_serial_clk0 / tx_serial_clk_1	In	N/A	The high speed serial clock generated by the PLL. Note: For Gen3 x1 ONLY tx_serial_clk_1 is used.

Port	Direction	Clock Domain	Description
pipe_hclk_in	In	N/A	The 500 MHz clock used for the ASN block. This clock is generated by the PLL, configured for Gen1/Gen2. Note: For Gen3 designs, use from the fPLL that is used for Gen1/Gen2.
pipe_hclk_out	Out	N/A	The 500 MHz clock output provided to the PHY - MAC interface.
PIPE Input from PHY - MAC Layer			
tx_parallel_data[31:0], [15:0] or [7:0]	In	tx_pma_clk	The TX parallel data driven from the MAC. For Gen1 this can be 8 or 16 bits. For Gen2 this is 16 bits, only. For Gen3 this is 32 bits. Note: unused_tx_parallel_data should be tied to '0'. Active High
tx_datak[3:0], [1:0] or [0]	In	tx_pma_clk	The data and control indicator for the received data. For Gen1 or Gen2, when 0, indicates that tx_parallel_data is data, when 1, indicates that tx_parallel_data is control. For Gen3, Bit[0] corresponds to pipe_txdata[7:0], bit[1] corresponds to pipe_txdata[15:8], and so on. Active High

Port	Direction	Clock Domain	Description
pipe_tx_sync_hdr[1:0]	In	tx_pma_clk	For Gen3, indicates whether the 130-bit block transmitted is a Data or Control Ordered Set Block. The following encodings are defined: 2'b10: Data block 2'b01: Control Ordered Set Block This value is read when pipe_tx_blk_start = 1b'1 Refer to *Lane Level Encoding" in the PCI Express Base Specification, Rev. 3.0* for a detailed explanation of data transmission and reception using 128b/130b encoding and decoding. Not used for Gen1 and Gen2 data rates. Active High
pipe_tx_blk_start	In	tx_pma_clk	For Gen3, specifies the start block byte location for TX data in the 128-bit block data. Used when the interface between the PCS and PHY-MAC (FPGA Core) is 32 bits. Not used for Gen1 and Gen2 data rates. Active High
pipe_tx_elecidle	In	tx_pma_clk	Forces the transmit output to electrical idle. Refer to *Intel PHY Interface for PCI Express (PIPE)* for timing diagrams. Active High

Port	Direction	Clock Domain	Description
pipe_tx_detectrx_loopback	In	tx_pma_clk	Instructs the PHY to start a receive detection operation. After power-up, asserting this signal starts a loopback operation. Refer to section 6.4 of *Intel PHY Interface for PCI Express (PIPE)* for a timing diagram. Active High
pipe_tx_compliance	In	tx_pma_clk	Asserted for one cycle to set the running disparity to negative. Used when transmitting the compliance pattern. Refer to section 6.11 of *Intel PHY Interface for PCI Express (PIPE) Architecture* for more information. Active High
pipe_rx_polarity	In	N/A	When 1'b1, instructs the PHY layer to invert the polarity on the received data. Active High
pipe_powerdown[1:0]	In	tx_pma_clk	Requests the PHY to change its power state to the specified state. The Power States are encoded as follows: 2'b00: P0 - Normal operation 2'b01: P0s - Low recovery time, power saving state 2'b10: P1 - Longer recovery time, lower power state 2'b11: P2 - Lowest power state

Port	Direction	Clock Domain	Description
pipe_tx_margin[2:0]	In	tx_pma_clk	Transmit V_{OD} margin selection. The PHY-MAC sets the value for this signal based on the value from the Link Control 2 Register. The following encodings are defined: 3'b000: Normal operating range 3'b001: Full swing: 800 - 1200 mV; Half swing: 400 - 700 mV 3'b010:-3'b011: Reserved 3'b100-3'b111: full swing: 200 - 400mV, half swing: 100 - 200 mV else reserved
pipe_tx_swing	In	tx_pma_clk	Indicates whether the transceiver is using full- or low-swing voltages as defined by the pipe_tx_margin 1'b0-Full swing 1'b1-Low swing
pipe_tx_deemph	In	tx_pma_clk	Transmit de-emphasis selection. In PCI Express Gen2 (5 Gbps) mode it selects the transmitter de-emphasis: 1'b0: -6 dB 1'b1: -3.5 dB

Port	Direction	Clock Domain	Description
pipe_g3_txdeemph[17:0]	In	tx_pma_clk	For Gen3, selects the transmitter de-emphasis. The 18 bits specify the following coefficients: [5:0]: C-1 [11:6]: C0 [17:12]: C+1 In Gen3 capable designs, the TX de-emphasis for Gen2 data rate is always -6 dB. The TX de-emphasis for Gen1 data rate is always -3.5 dB. Refer to section 6.6 of *Intel PHY Interface for PCI Express (PIPE) Architecture* for more information.
pipe_g3_rxpresethint	In	tx_pma_clk	Provides the RX preset hint for the receiver.
pipe_rx_eidlein-fersel[2:0]	In	tx_pma_clk	When asserted high, the electrical idle state is inferred instead of being identified using analog circuitry to detect a device at the other end of the link. The following encodings are defined: 3'b0xx: Electrical Idle Inference not required in current LTSSM state 3'b100: Absence of COM/SKP OS in 128 ms window for Gen1 or Gen2 3'b101: Absence of TS1/TS2 OS in 1280 UI interval for Gen1 or Gen2 3'b110: Absence of Electrical Idle Exit in 2000 UI interval for Gen1 and 16000 UI interval for Gen2 3'b111: Absence of Electrical Idle exit in 128 ms window for Gen1

Port	Direction	Clock Domain	Description
pipe_rate[1:0]	In	N/A	The 2-bit encodings defined in the following list: 2'b00: Gen1 rate (2.5 Gbps) 2'b01: Gen2 rate (5.0 Gbps) 2'b1x: Gen3 (8.0 Gbps)
pipe_sw_done	In	N/A	Signal from the Master clock generation buffer, indicating that the rate switch has completed. Use this signal for bonding mode only. For non-bonded applications, this signal is internally connected to the local CGB.
pipe_tx_data_valid	In	tx_pma_clk	For Gen3, this signal is de-asserted by the MAC to instruct the PHY to ignore tx_parallel_data for one clock cycle. A value of 1'b1 indicates the PHY should use the data. A value of 0 indicates the PHY should not use the data. Active High
PIPE Output to PHY - MAC Layer			
rx_parallel_data[31:0], [15:0] or [7:0]	Out	tx_pma_clk	The RX parallel data driven to the MAC. For Gen1 this can be 8 or 16 bits. For Gen2 this is 16 bits only. For Gen3 this is 32 bits.
rx_datak[3:0], [1:0] or [0]	Out	tx_pma_clk	The data and control indicator. For Gen1, when 0, indicates that tx_parallel_data is data, when 1, indicates that tx_parallel_data is control. For Gen3, Bit[0] corresponds to pipe_txdata[7:0], Bit[1] corresponds to pipe_txdata[15:8], and so on.

Port	Direction	Clock Domain	Description
pipe_rx_sync_hdr[1:0]	Out	tx_pma_clk	For Gen3, indicates whether the 130-bit block being transmitted is a Data or Control Ordered Set Block. The following encodings are defined: 2'b10: Data block 2'b01: Control Ordered Set block This value is read when rx_blk_start = 4'b0001. Refer to *Section 4.2.2.1. Lane Level Encoding* in the *PCI Express Base Specification, Rev. 3.0* for a detailed explanation of data transmission and reception using 128b/130b encoding and decoding.
pipe_rx_blk_start	Out	tx_pma_clk	For Gen3, specifies the start block byte location for RX data in the 128-bit block data. Used when the interface between the PCS and PHYMAC (FPGA Core) is 32 bits. Not used for Gen1 and Gen2 data rates. Active High
pipe_rx_data_valid	Out	tx_pma_clk	For Gen3, this signal is de-asserted by the PHY to instruct the MAC to ignore rx_parallel_data for one clock cycle. A value of 1'b1 indicates the MAC should use the data. A value of 1'b0 indicates the MAC should not use the data. Active High
pipe_rx_valid	Out	tx_pma_clk	Asserted when RX data and control are valid.
pipe_phy_status	Out	tx_pma_clk	Signal used to communicate completion of several PHY requests. Active High

Port	Direction	Clock Domain	Description
pipe_rx_elecidle	Out	N/A	When asserted, the receiver has detected an electrical idle. Active High
pipe_rx_status[2:0]	Out	tx_pma_clk	Signal encodes receive status and error codes for the receive data stream and receiver detection. The following encodings are defined: 3'b000 - receive data OK 3'b001 - 1 SKP added 3'b010 - 1 SKP removed 3'b011 - Receiver detected 3'b100 - Both 8B/10B or 128b/130b decode error and (optionally) RX disparity error 3'b101 - Elastic buffer overflow 3'b110 - Elastic buffer underflow 3'b111 - Receive disparity error, not used if disparity error is reported using 3'b100.
pipe_sw	Out	N/A	Signal to clock generation buffer indicating the rate switch request. Use this signal for bonding mode only. For non-bonded applications this signal is internally connected to the local CGB. Active High

How to Place Channels for PIPE Configurations

All the placement restrictions are dictated by the hardware, not by the fitter (or software model). The restrictions are listed below:

1. The channels must be contiguous for bonded designs.
2. The master CGB is the only way to access x6 lines and has to be used in bonded designs. ie. the local CGB cannot be used to route clock signals to slave channels (the local CGB does not have access to x6 lines)

Master Channel in Bonded Configurations

For PCIe, both the PMA and PCS must be bonded. There is no need to specify the PMA Master Channel because of the separate Master CGB in the hardware. However, a logical PCS Master Channel must be

specified and the Native PHY provides an option to set this. You can choose any one of the data channels (part of the bonded group) as the logical PCS master channel.

Note: Whichever channel you pick as the PCS master, the fitter will select physical CH1 or CH4 of a 6-pack as the master channel. This is because the ASN and Master CGB connectivity only exists in the hardware of these two channels of the 6-pack.

Altera recommends the following as defaults:

PIPE Configuration	Logical PCS Master Channel # (default)
x1	1
x2	2
x4	2
x8	5

The following figures show the default configurations:

Figure 2-85: x2 Configuration

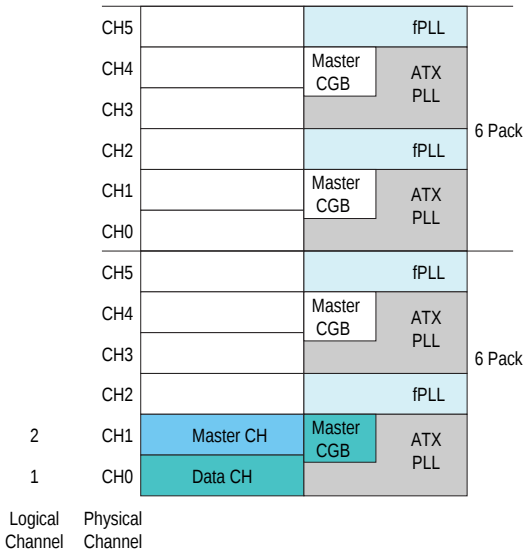


Figure 2-86: x4 Configuration

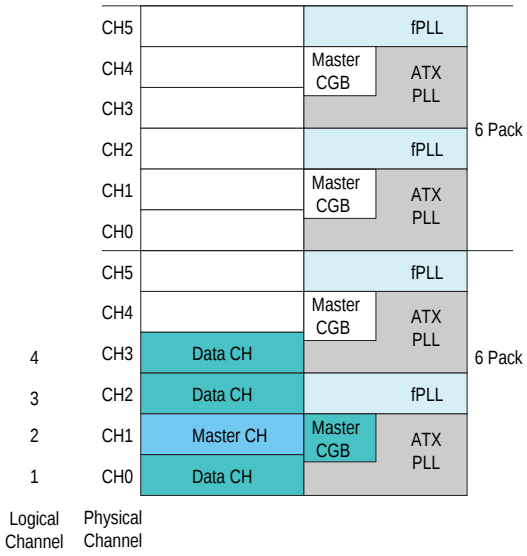


Figure 2-87: x8 Configuration

For x8 configurations, Altera recommends you choose a master channel that is a maximum of four channels away from the farthest slave channel.

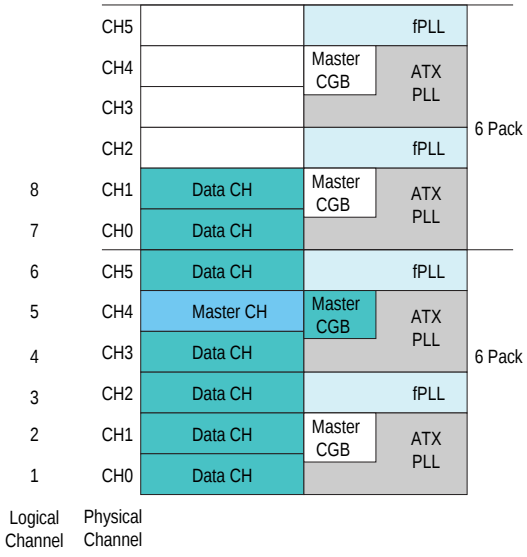
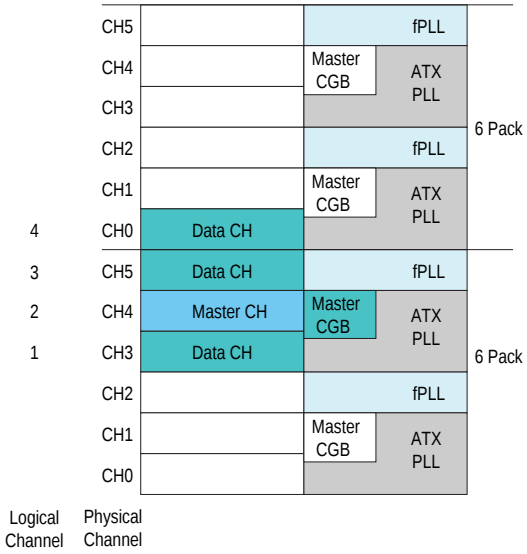


Figure 2-88: x4 Alternate Configuration

The figure below shows an alternate way of placing 4 bonded channels. In this case, the logical PCS master channel number must be specified as 2.



As indicated in the figures above, the fitter picks either physical CH1 or CH4 as the PCS master in bonded configurations for PIPE.

Design Example

The *PIPE Design Example*, located on the Arria 10 Transceiver PHY Design Examples Wiki page, demonstrates the connectivity between several IPs that form a complete PCIe design. The example contains the following components:

- **PHY**—Native PHY IP configured for PIPE Gen3 x8 mode
- **ATX PLL**—PLL used for Gen3 data rate
- **fPLL**—PLL used for Gen1 and Gen2 data rates
- **Reset controller**
- **MAC**
- **Data generator**

The design example exercises the PIPE-specific features and blocks. The pseudo-MAC exercises the control signals and implements part of the LTSSM. The data generator exercises various digital decoding blocks. The data generator and checker can generate and verify various ordered sets such as TS1, TS2, EIOS, EIEOS, and SKP OS, as well as scrambling and descrambling data while operating at Gen3 rates.

The *PIPE Design Example User Guide*, located in the PIPE Design File on the Wiki page, contains recommendations about SDC timing constraints.

Note: The design examples on the Wiki page provide useful guidance for developing your own designs, but they are not guaranteed by Altera. Use them with caution.

Related Information

PIPE Design Example

CPRI

The CPRI interface is a high-speed serial interface developed for wireless network REC to uplink and downlink data from available remote RE.

The CPRI protocol defines the interface of radio base stations between the radio equipment controller (REC) and the radio equipment (RE). The physical layer supports both the electrical interfaces (for example, traditional radio base stations) and the optical interface (for example, radio base stations with a remote radio head). The scope of the CPRI specification is restricted to the link interface only, which is a point-to-point interface. The link has all the features necessary to enable a simple and robust usage of any given REC and RE network topology, including a direct interconnection of multiport REs.

Transceiver Channel Datapath and Clocking for CPRI

You can accurately compute the transceiver datapath latencies when using the CPRI protocol standardized interfaces.

Figure 2-89: Transceiver Channel Datapath and Clocking for CPRI

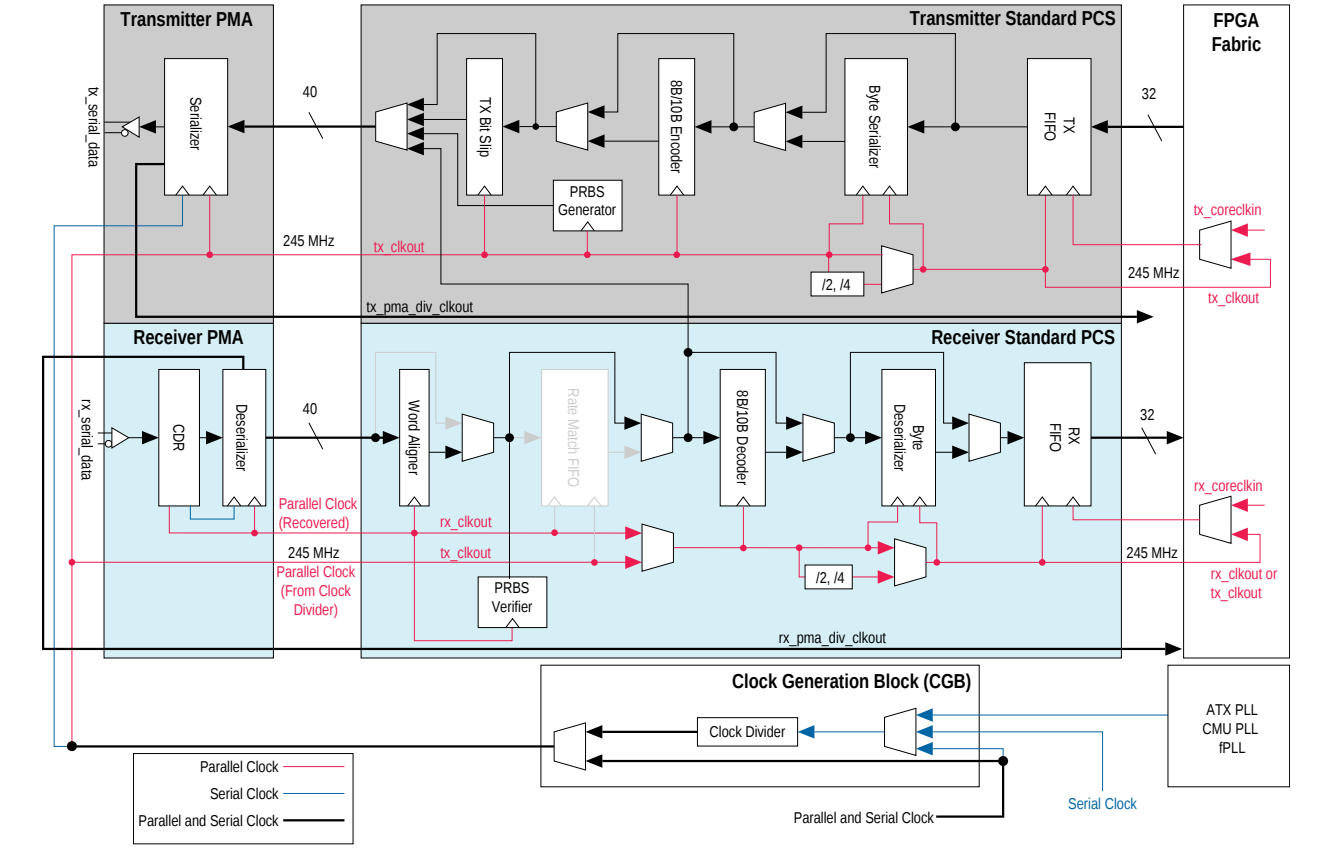


Table 2-132: Channel Width Options for Supported Serial Data Rates

Serial Data Rate (Mbps)	Channel Width (FPGA-PCS Fabric)			
	8/10 Bit Width		16/20 Bit Width	
	8-Bit	16-Bit	16-Bit	32-Bit
614.4	Yes	N/A	N/A	N/A
1228.8	Yes	Yes	Yes	Yes
2457.6	N/A	Yes	Yes	Yes
3072	N/A	Yes	Yes	Yes
4915.2	N/A	N/A	N/A	Yes
6144	N/A	N/A	N/A	Yes
9830.4	N/A	N/A	N/A	Yes

TX PLL Selection for CPRI

Choose a transmitter PLL that fits your required data rate.

Table 2-133: TX PLL Supported Data Rates

TX PLLs	Supported Data Rate (Mbps)
ATX	1228.8, 2457.6, 3072, 4915.2, 6144, 9830.4
fPLL	1228.8, 2457.6, 3072, 4915.2, 6144
CMU	1228.8, 2457.6, 3072, 4915.2, 6144

Auto-Negotiation

When auto-negotiation is required, the channels initialize at the highest supported frequency and switch to successively lower data rates if frame synchronization is not achieved. If your design requires auto-negotiation, choose a base data rate that minimizes the number of PLLs required to generate the clocks required for data transmission.

By selecting an appropriate base data rate, you can change data rates by changing the local clock generation block (CGB) divider. If a single base data rate is not possible, you can use an additional PLL to generate the required data rates.

Table 2-134: Recommended Base Data Rates and Clock Generation Blocks for Available Data Rates

Data Rate (Mbps)	Base Data Rate	Local CGB Divider
1228.8	9830.4	8
2457.6	9830.4	4
3072.0	6144.0	2
4915.2	9830.4	2
6144.0	6144.0	1

Data Rate (Mbps)	Base Data Rate	Local CGB Divider
9830.4	9830.4	1

Supported Features for CPRI

The CPRI protocol places stringent requirements on the amount of latency variation that is permissible through a link that implements these protocols.

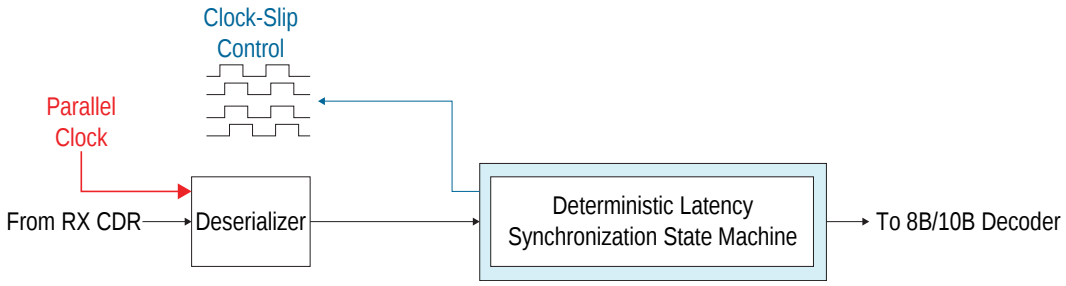
CPRI (Auto) and CPRI (Manual) transceiver configuration rules are both available for CPRI designs. Both modes use the same functional blocks, but the configuration mode of the word aligner is different between the Auto and Manual modes. In CPRI (Auto) mode, the word aligner works in deterministic mode. In CPRI (manual) mode, the word aligner works in manual mode.

To avoid transmission interference in time division multiplexed systems, every radio in a cell network requires accurate delay estimates with minimal delay uncertainty. Lower delay uncertainty is always desired for increased spectrum efficiency and bandwidth. The devices are designed with features to minimize the delay uncertainty for both RECs and REs.

Word Aligner in Deterministic Latency Mode for CPRI

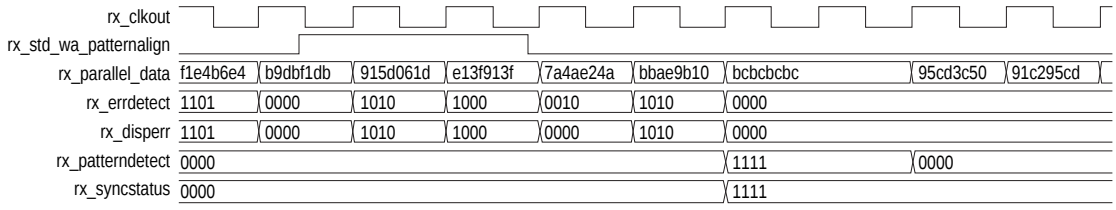
The deterministic latency state machine in the word aligner reduces the known delay variation from the word alignment process. It automatically synchronizes and aligns the word boundary by slipping a clock cycle in the deserializer. Incoming data to the word aligner is aligned to the boundary of the word alignment pattern (K28.5).

Figure 2-90: Deterministic Latency State Machine in the Word Aligner



When using deterministic latency state machine mode, reassert `rx_std_wa_patternalign` to initiate the pattern alignment after the initial alignment following the deassertion of reset.

Figure 2-91: Word Aligner in Deterministic Mode Waveform



Related Information

Word Aligner on page 5-37

Transmitter and Receiver Latency

The latency variation from the link synchronization function (in the word aligner block) is deterministic with the `rx_bitslipboundaryselectout` port. In additional, you can optionally fix the round trip transceiver latency for port implementation in the remote radio head to compensate for latency variation in the word aligner block with the `tx_bitslipboundaryselect` port. The `tx_bitslipboundaryselect` port is available to control the amount of bits to be slipped in the transmitter serial data stream. You can optionally use the `tx_bitslipboundaryselect` port to round the round-trip latency to a whole number of cycles.

To use the byte deserializer, create additional logic in the FPGA fabric to determine if the comma byte is received in the lower or upper byte of the word. The delay is dependent on the word in which the comma byte appears.

The total transmitter and receiver channel datapath latencies are computed as follows:

- The total transmitter channel datapath latency is equal to the transmitter fixed latency and `tx_bitslipboundaryselect` delay.
- The total receiver channel datapath latency is equal to the receiver fixed latency, `rx_std_bitslipboundarysel` delay, and byte deserializer delay.

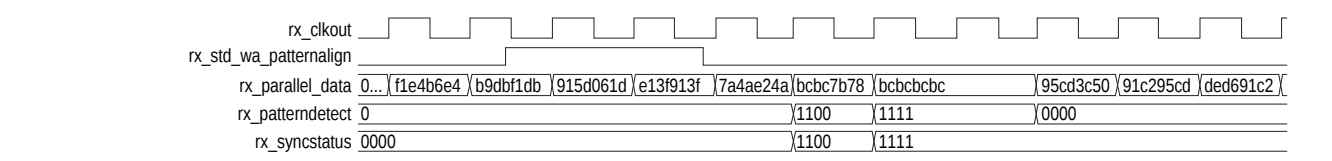
Note: Latency numbers are pending device characterization.

Word Aligner in Manual Mode for CPRI

When configuring the word aligner in CPRI (Manual), the word aligner parses the incoming data stream for a specific alignment character.

After `rx_digitalreset` deasserts, asserting the `rx_std_wa_patternalign` triggers the word aligner to look for the predefined word alignment pattern or its complement in the received data stream, and automatically synchronizes to the new word boundary. Any alignment pattern found thereafter in a different word boundary causes the word aligner to resynchronize to this new word boundary if the `rx_std_wa_patternalign` remains asserted. If you deassert `rx_std_wa_patternalign`, the word aligner maintains the current word boundary even when it finds the alignment pattern in a new word boundary. When the word aligner is synchronized to the new word boundary, `rx_patterndetect` and `rx_syncstatus` are assert for one parallel clock cycle.

Figure 2-92: Word Aligner in Manual Alignment Mode Waveform



Related Information

Word Aligner on page 5-37

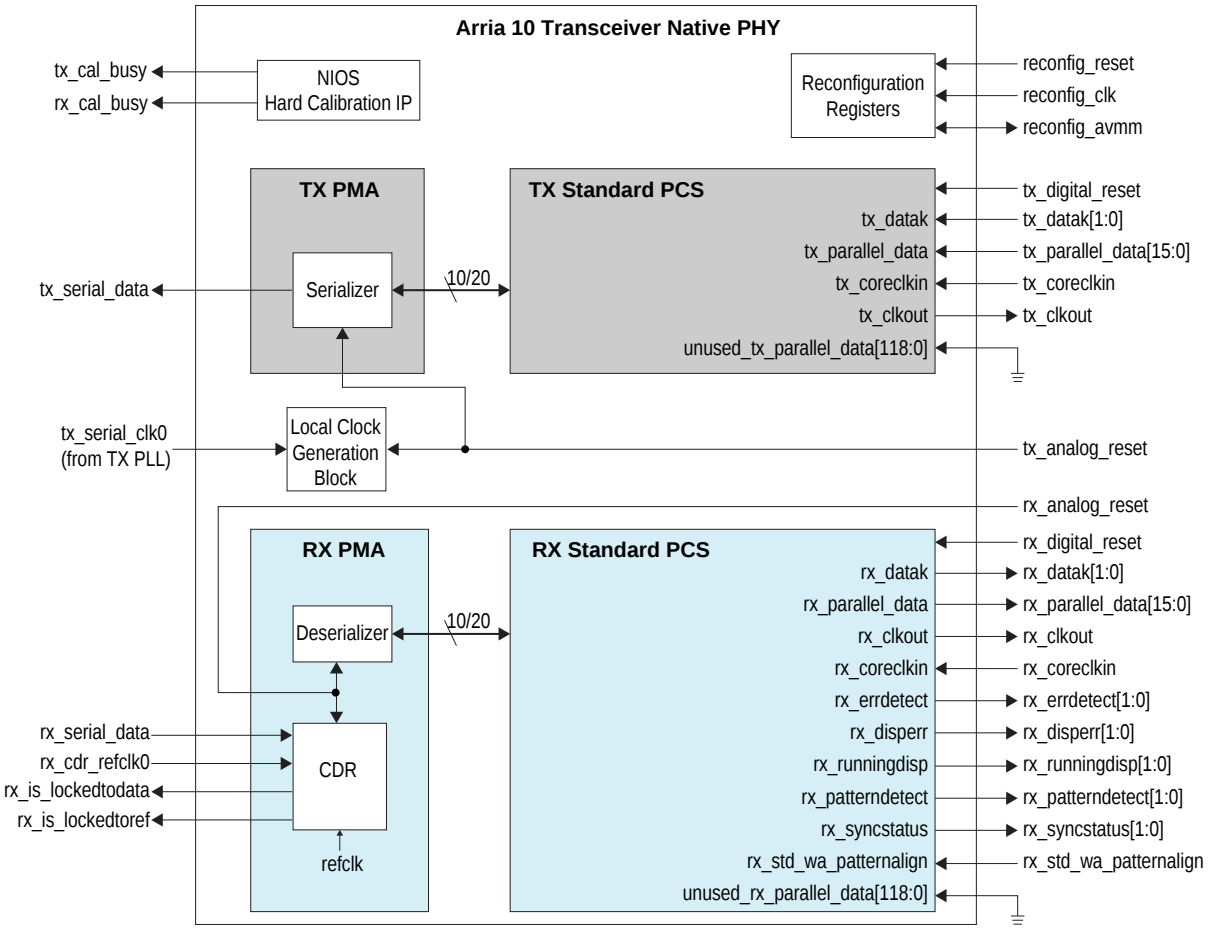
How to Implement CPRI in Arria 10 Transceivers

Before you begin

You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing your CPRI protocol.

1. Open the MegaWizard Plug-In Manager and select the PHY IP.
   Refer to **Select and Instantiate PHY IP** on page 2-2.
2. Select **CPRI (Auto)** or **CPRI (Manual)** from the **Transceiver configuration rules** list located under **Datapath Options**, depending on which protocol you are implementing.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for CPRI** on page 2-222 as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the setting to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 2-93: Signals and Ports of Native PHY IP for CPRI

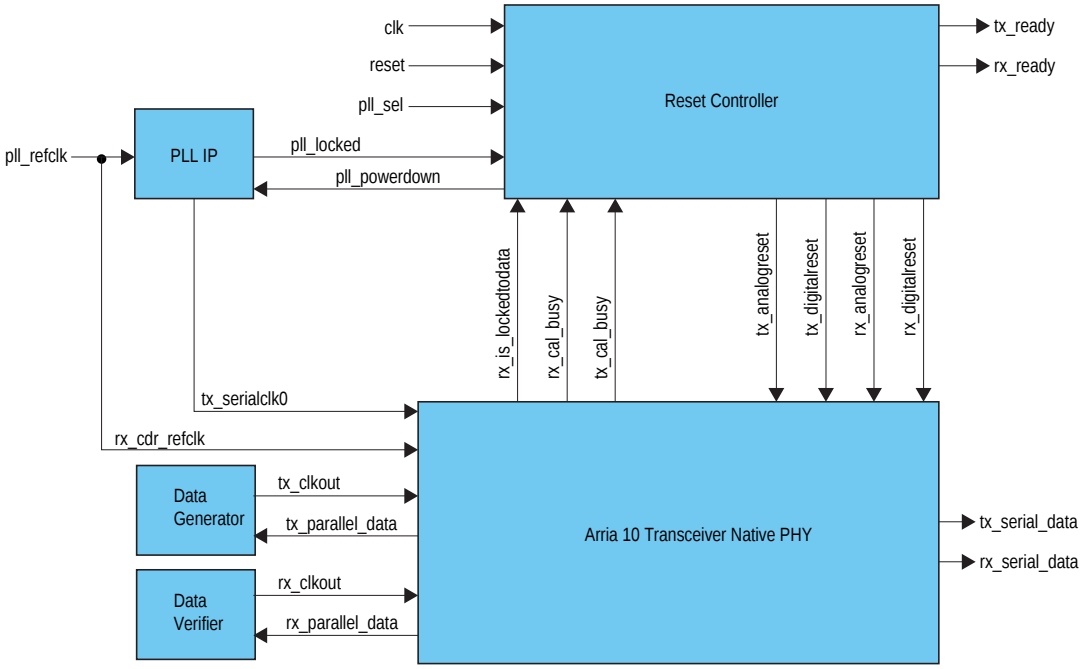


5. Instantiate and configure your PLL.
6. Create a transceiver reset controller.

You can use your own reset controller or use the Native PHY Reset Controller IP.

7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in **Figure 2-93** to connect the ports.

Figure 2-94: Connection Guidelines for a CPRI PHY Design



8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Standard PCS Architecture** on page 5-31
  For more information about Standard PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Standard PCS and PMA Ports** on page 2-52
  Port definitions for the Transceiver Native PHY Standard Datapath

Native PHY IP Parameter Settings for CPRI

Table 2-135: General and Datapath Options

The first two sections of the MegaWizard Plug-In Manager for the Native PHY IP provide a list of general and datapath options to customize the transceiver.

Parameter	Value
Device speed grade	fastest
Message level for rule violations	error message
Transceiver configuration rules	• CPRI (Auto) • CPRI (Manual)
Transceiver mode	TX/RX Duplex
Number of data channels	1
Data rate	1.2288 Gbps 9.830 Gbps
Enable reconfiguration between Standard and Enhanced PCSs	Off
Enable simplified data interface	On

Table 2-136: TX PMA Parameters

Parameter	Value
TX channel bonding mode	Not bonded
TX local clock division factor	1
Number of TX PLL clock inputs per channel	1
Initial TX PLL clock input selection	0
Enable tx_pma_clkout port	Off
Enable tx_pma_div_clkout port	Off
tx_pma_div_clkout division factor	Disabled
Enable tx_pma_elecidle port	Off
Enable tx_pma_qpipullup port (QPI)	Off
Enable tx_pma_qpipulldn port (QPI)	Off
Enable tx_pma_txdetectrx port (QPI)	Off
Enable tx_pma_rxfound port (QPI)	Off

Table 2-137: RX PMA Parameters

Parameter	Value
Number of CDR reference Clocks	1
Selected CDR reference clock	0
Selected CDR reference clock frequency	Select legal range defined by the Quartus II software
PPM detector threshold	1000
Decision feedback equalization mode	Disabled
Enable rx_pma_clkout port	Off
Enable rx_pma_div_clkout port	Off
rx_pma_div_clkout division factor	Disabled
Enable rx_pma_clkslip port	Off
Enable rx_pma_qpipulldn port (QPI)	Off
Enable rx_is_lockedtodata port	Off
Enable rx_is_lockedtoref port	Off
Enable rx_set_locktodata and rx_set_locktoref ports	Off
Enable rx_seriallpbken port	Off
Enable PRBS verifier control and status ports	Off

Table 2-138: Standard PCS Parameters

Parameters	Value
Standard PCS / PMA interface width	10/20
Enable Standard PCS low latency mode	Off
TX FIFO mode	register_fifo
RX FIFO Mode	register_fifo
Enable tx_std_pcfifo_full port	Off
Enable tx_std_pcfifo_empty port	Off
Enable rx_std_pcfifo_full	Off
Enable rx_std_pcfifo_empty port	Off
TX byte serializer mode	• Disabled • Serialize x2
RX byte deserializer mode	• Disabled • Deserialize x2

Parameters	Value
Enable TX 8B/10B encoder	**On**
Enable TX 8B/10B disparity control	**Off**
Enable RX 8B/10B decoder	**On**
RX rate match FIFO mode	Disabled
RX rate match insert / delete -ve pattern (hex)	`0x00000000`
RX rate match insert / delete +ve pattern (hex)	`0x00000000`
Enable rx_std_rmfifo_full port	**Off**
Enable rx_std_rmfifo_empty port	**Off**
PCI Express Gen 3 rate match FIFO mode	**Bypass**
Enable TX bit slip	**On**
Enable tx_std_bitslipboundarysel port	**On**
RX word aligner mode	• **Deterministic latency** (CPRI Auto configuration) • **Manual (PLD controlled)** (CPRI Manual configuration)
RX word aligner pattern length	**10**
RX word aligner pattern (hex)	`0x17c`
Number of word alignment patterns to achieve sync	**3**
Number of invalid data words to lose sync	**3**
Number of valid data words to decrement error count	**3**
Enable rx_std_wa_patternalign port	**Off**
Enable rx_std_wa_a1a2size port	**Off**
Enable rx_std_bitslipboundarysel port	**Off**
Enable rx_bitslip port	• **Off**(CPRI Auto configuration) • **On** (CPRI Manual configuration)
All options under **Word Bit Reversal and Polarity Inversion**	**Off**
All options under **PCIe Ports**	**Off**

Table 2-139: Dynamic Reconfiguration

Parameter	Value
Enable dynamic reconfiguration	**Off**
Share reconfiguration interface	**Off**
Enable embedded JTAG AVMM master	**On/Off**

Parameter	Value
Configuration file prefix	altera_xcvr_native_a10
Generate SystemVerilog package file	**On/Off**
Generate C header file	**On/Off**
Generate MIF (Memory Intialization File)	**On/Off**

Table 2-140: General Options

Parameter	Value
Generate parameter documentation file	**On/Off**

Other Protocols

Using the "Basic" and "Basic with KR FEC" Configurations of Enhanced PCS

You can use Arria 10 transceivers to configure the Enhanced PCS to support other 10G or 10G-like protocols. The Basic (Enhanced PCS) transceiver configuration rule allows access to the Enhanced PCS with full user control over the transceiver interfaces, parameters, and ports.

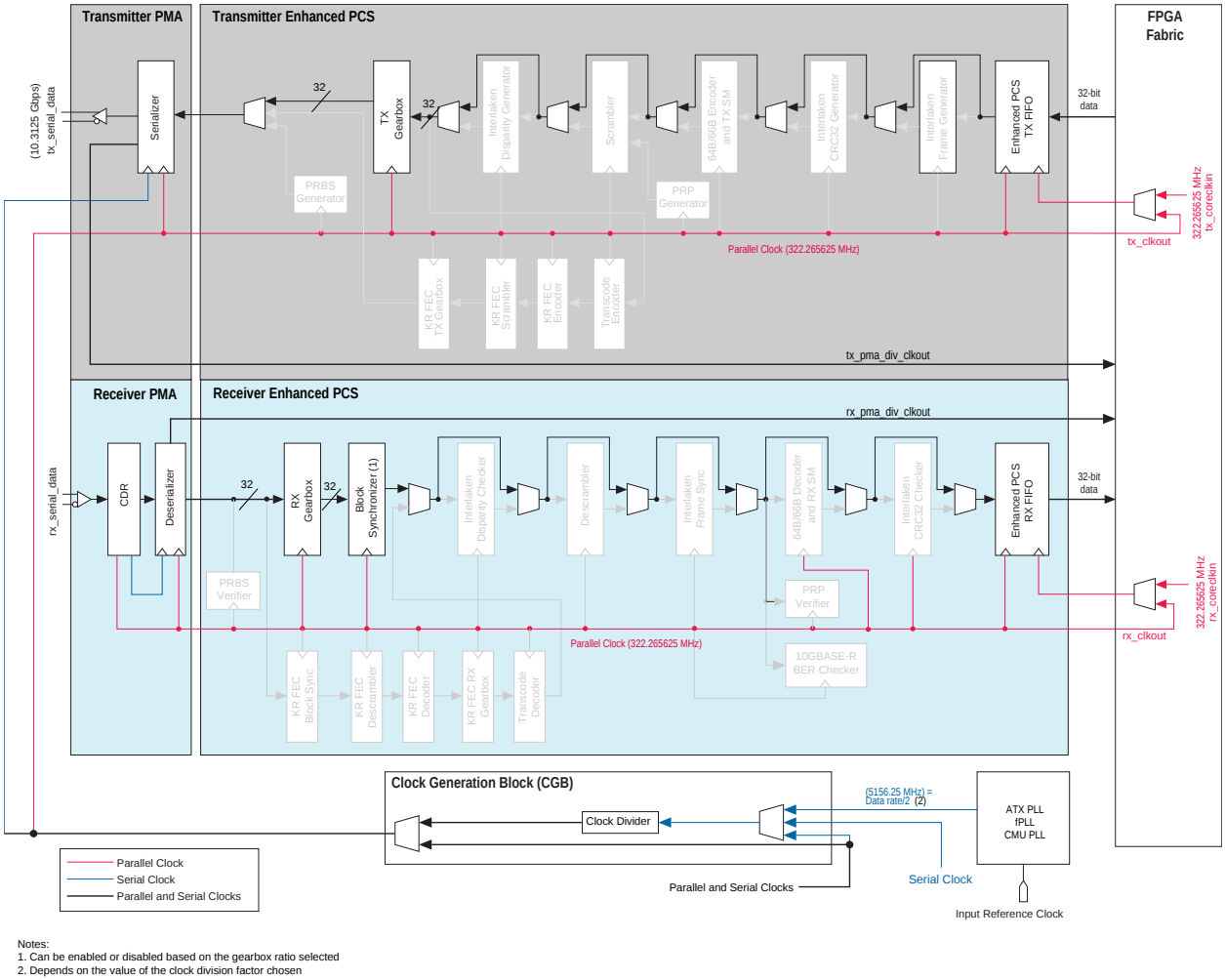
You can configure the transceivers for Basic functionality using the **Native PHY IP Basic (Enhanced PCS)** transceiver configuration rule.

Basic with KR FEC is a KR FEC sublayer support with a low latency physical coding sublayer (PCS). The KR FEC sublayer increases the bit error rate (BER) performance of a link. Use this configuration to implement applications with low latency or low BER requirements or applications such as 10 Gbps or 40 Gbps Ethernet over backplane (10GBASER-KR or 40GBASE-KR4 protocol).

The Forward Error Correction (FEC) function is defined in Clause 74 of IEEE 802.3ap-2007. FEC provides an error detection and correction mechanism that allows noisy channels to achieve the Ethernet-mandated Bit Error Rate (BER) of 10^{-12}. The FEC sublayer provides additional link margin by compensating for variations in manufacturing and environmental conditions. To distinguish it from other FEC mechanisms (for example, Optical Transport Network FEC), FEC as defined in Clause 74 of IEEE 802.3ap-2007 is called KR FEC.

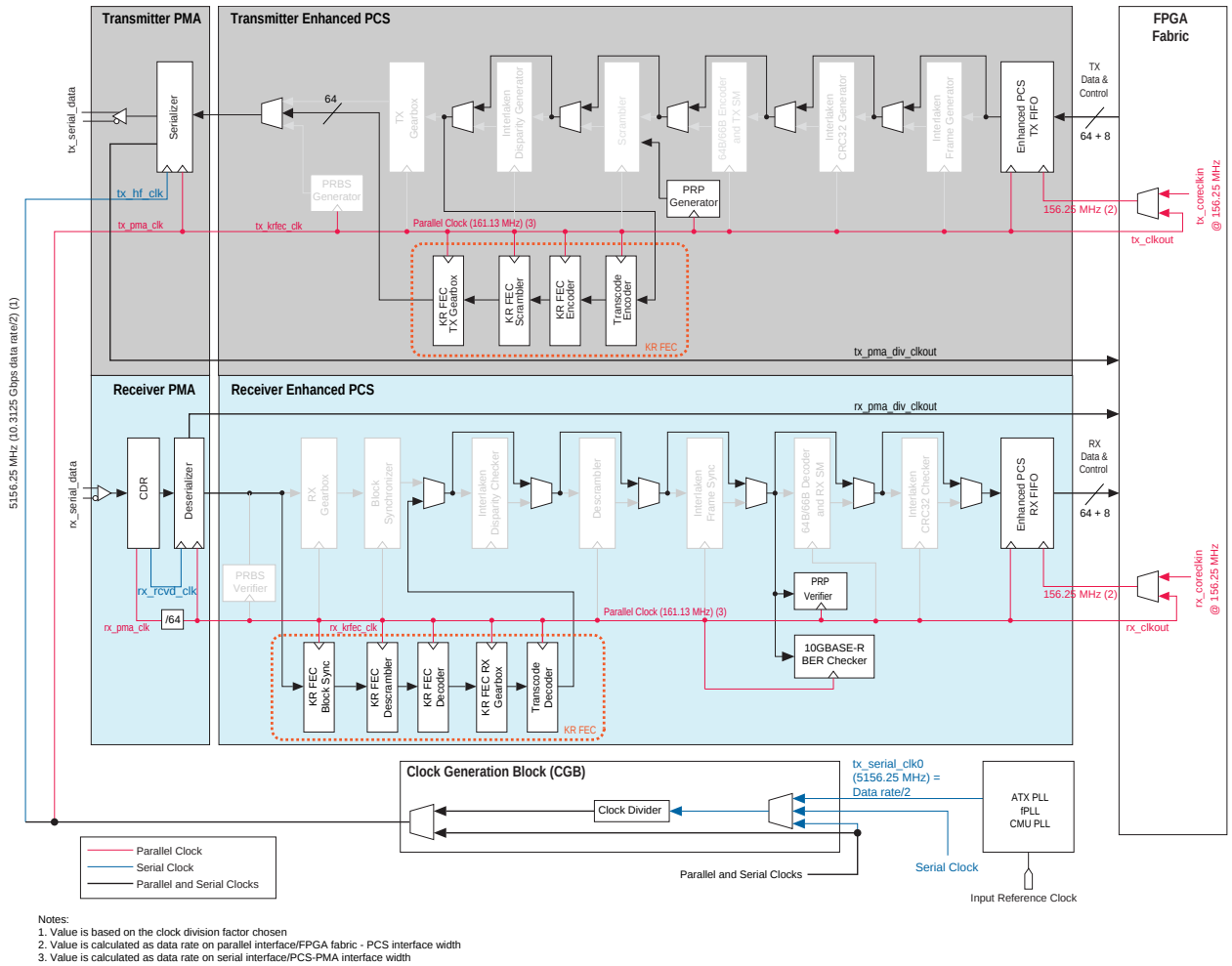
Note: This configuration supports the FIFO in phase compensation and register modes, and KR FEC PCS blocks. You can implement all other required logic for your specific application, such as standard or proprietary protocol multi-channel alignment, either in the FPGA fabric in soft IP or use Altera's 10GBASE-KR PHY or 40GBASE-KR4 PHY IP core products as full solutions in the FPGA.

Figure 2-95: Transceiver Channel Datapath and Clocking for Basic (Enhanced PCS) Configuration



Notes:
1. Can be enabled or disabled based on the gearbox ratio selected
2. Depends on the value of the clock division factor chosen

Figure 2-96: Transceiver Channel Datapath and Clocking for a Basic with KR FEC Configuration



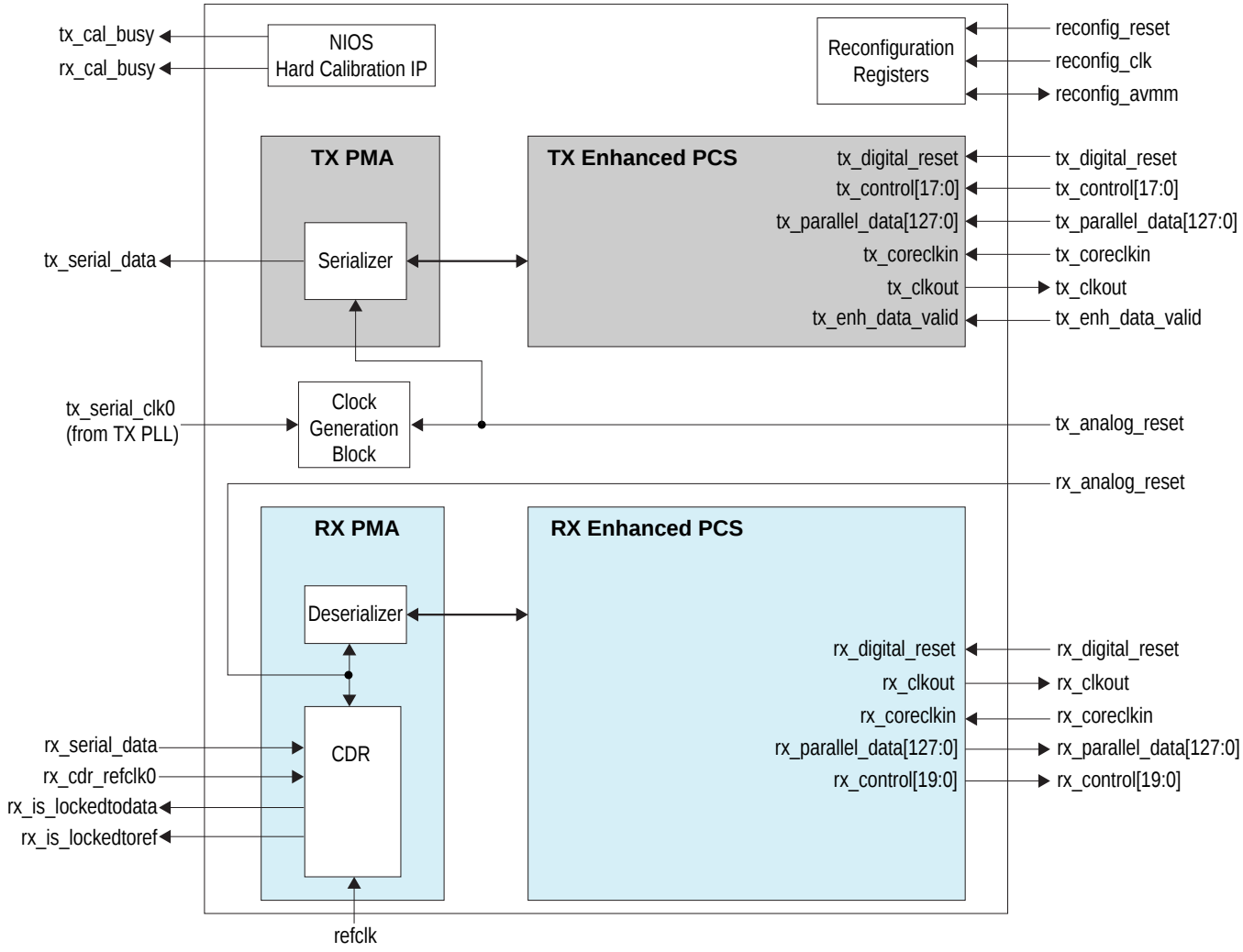
How to Implement the Basic (Enhanced PCS) and Basic with KR FEC Transceiver Configuration Rules in Arria 10 Transceivers

Before you begin

You should be familiar with the Basic (Enhanced PCS) and PMA architecture, PLL architecture, and the reset controller before implementing the Basic (Enhanced PCS) or Basic with KR FEC Transceiver Configuration Rule.

1. Open the MegaWizard Plug-In Manager and select the **Arria 10 Transceiver Native PHY IP**. Refer to **Select and Instantiate PHY IP** on page 2-2 for more details.
2. Select **Basic (Enhanced PCS), Basic with KR FEC** from the **Transceiver Configuration Rule** list located under **Datapath Options**.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for Basic (Enhanced PCS) and Basic with KR FEC** as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 2-97: Signals and Ports of Native PHY IP for Basic (Enhanced PCS), Basic with KR FEC Configurations



5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Altera Transceiver PHY Reset Controller IP.
7. Connect the Native PHY IP to the PLL IP and the reset controller.

Figure 2-98: Connection Guidelines for a Basic (Enhanced PCS) Transceiver Design

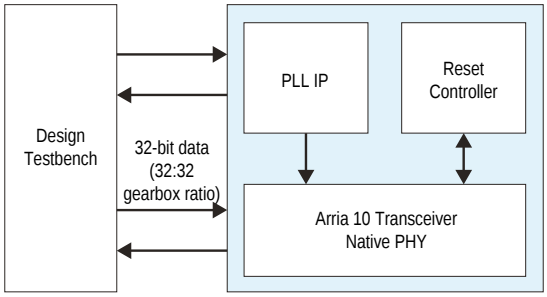
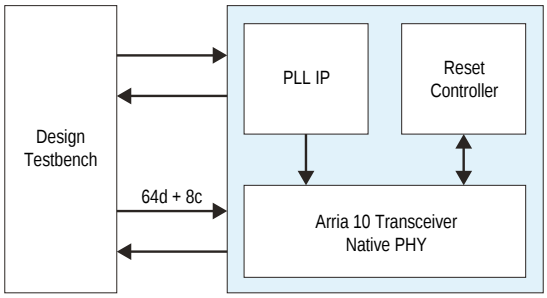


Figure 2-99: Connection Guidelines for a Basic with KR FEC Transceiver Design



8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Enhanced PCS Architecture** on page 5-14
  For more information about Enhanced PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Enhanced PCS and PMA Ports** on page 2-37
  For detailed information about the available ports in the Basic protocol.

Native PHY IP Parameter Settings for Basic (Enhanced PCS) and Basic with KR FEC

Table 2-141: General and Datapath Parameters

The first two sections of the MegaWizard Plug-In Manager for the Transceiver Native PHY provide a list of general and datapath options to customize the transceiver.

Parameter	Range
Device speed grade	**fastest**
Message level for rule violations	**error**, **warning**
Transceiver Configuration Rule	**Basic (Enhanced PCS)**, **Basic with KR FEC**
Transceiver mode	**TX / RX Duplex**, **TX Simplex**, **RX Simplex**
Number of data channels	**1** to **96**
Data rate	**125 Mbps** to **28.1 Gbps**
Enable reconfiguration between Standard and Enhanced PCS	**On / Off**
Enable simplified data interface	**On / Off**

Table 2-142: TX PMA Parameters

Parameter	Range
TX channel bonding mode	**Non bonded**, **PMA bonding**, **PMA/PCS bonding**
PCS TX channel bonding master	**Auto**, **0**, **1**
Actual PCS TX channel bonding master	**0**, **1**
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0**

Table 2-143: RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	For Basic (Enhanced PCS): Depends on the data rate parameter For Basic with KR FEC: **50** to **800**
PPM detector threshold	**62.5**, **100**, **125**, **200**, **250**, **300**, **500**, **1000**

Parameter	Range
Decision feedback equalization mode	Disabled, Fixed tap, Floating tap

Table 2-144: Enhanced PCS Parameters

Parameter	Range
Enhanced PCS/PMA interface width	32, 40, 64 **Note:** Basic with KR FEC allows 64 only
FPGA fabric/Enhanced PCS interface width	32, 40, 64, 66 **Note:** Basic with KR FEC allows 66 only
Enable RX/TX FIFO double-width mode	On / Off
TX FIFO mode	Phase Compensation, Register, Basic
TX FIFO partially full threshold	10, 11, 12, 13, 14, 15
TX FIFO partially empty threshold	1, 2, 3, 4, 5
RX FIFO mode	Phase Compensation, Register, Basic
RX FIFO partially full threshold	0 to 31
RX FIFO partially empty threshold	0 to 31

Table 2-145: Block Sync Parameters

Parameter	Range
Enable RX block synchronizer	On / Off

Table 2-146: Gearbox Parameters

Parameter	Range
Enable TX data bitslip	On / Off
Enable TX data polarity inversion	On / Off
Enable RX data bitslip	On / Off
Enable RX data polarity inversion	On / Off

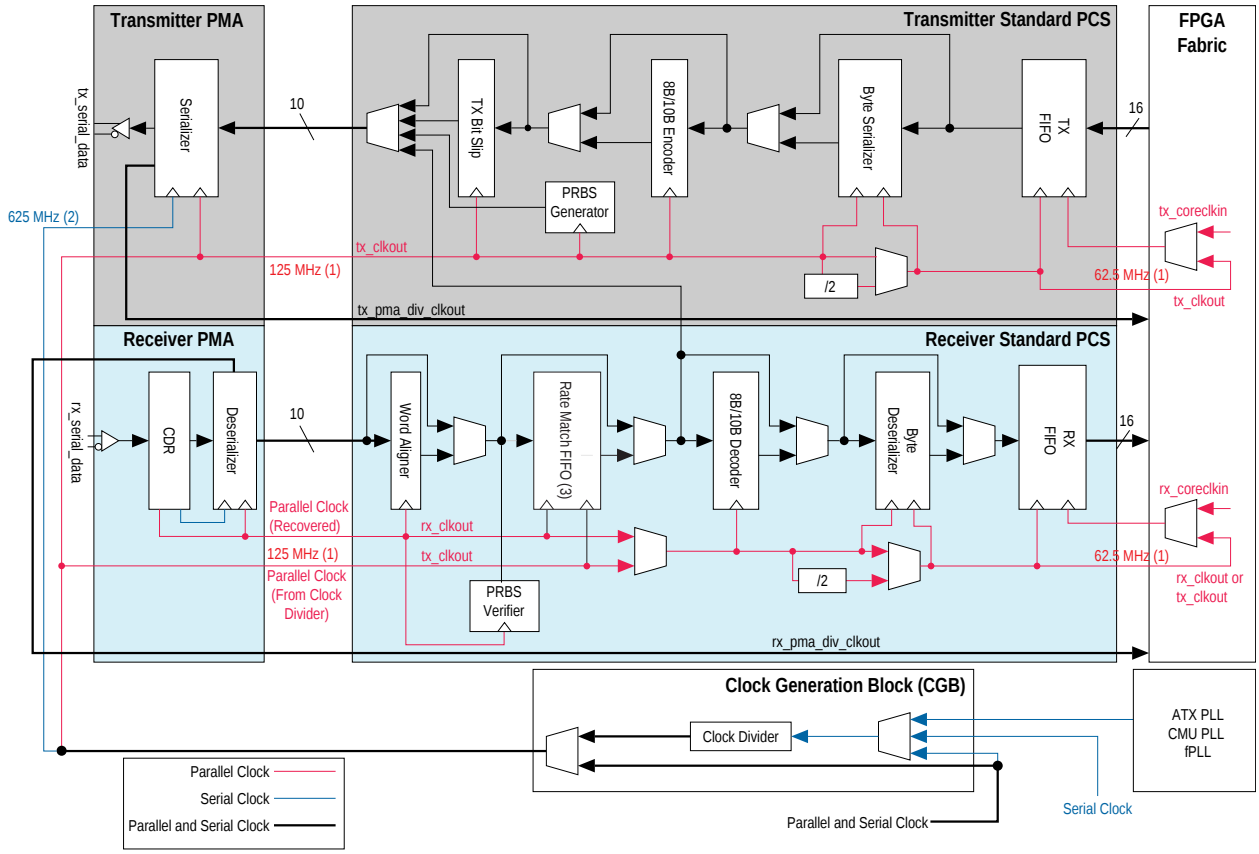
Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS

Use one of the following transceiver configuration rules to implement protocols such as SONET/SDH, SDI/HD, SATA, or your own custom protocol:

- Basic protocol
- Basic protocol with low latency enabled
- Basic with rate match protocol

Figure 2-100: Transceiver Channel Datapath and Clocking for the Basic and Basic with Rate Match Configurations

The data rate is 1250 Mbps.

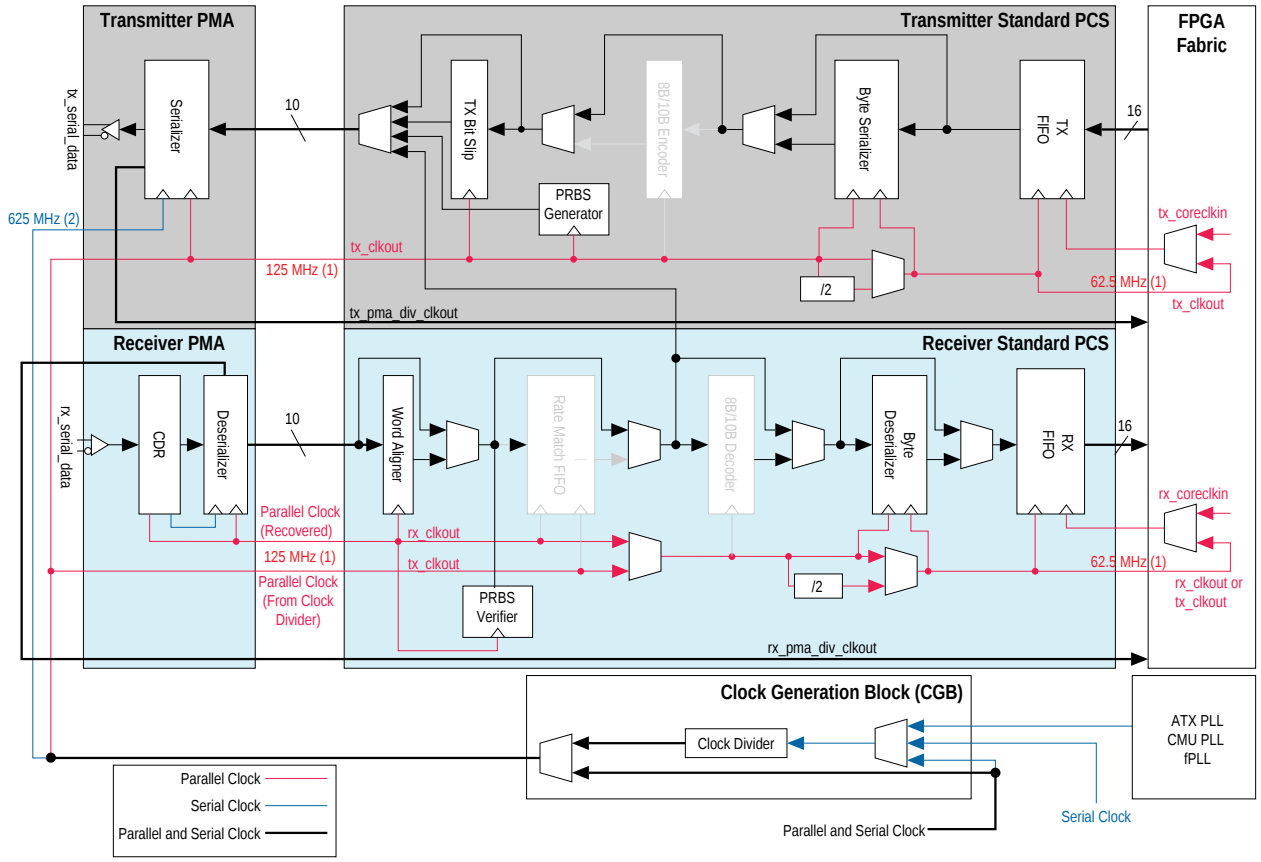


Notes:
1. The parallel clock (tx_clkout or rx_clkout) is calculated as data rate/PCS-PMA interface width =1250/10 = 125 MHz. When the Byte Serializer is set to Serialize x2 mode, tx_clkout and rx_clkout become 1250/20 = 62.5 MHz.
2. The serial clock is calculated as data rate/2. The PMA runs on a dual data rate clock.
3. This block is only enabled when using the Basic with Rate Match transceiver configuration rule.

In low latency mode, much of the Standard PCS is bypassed, which allows more design control in the FPGA fabric.

Figure 2-101: Transceiver Channel Datapath and Clocking for Basic Configuration with Low Latency Enabled

The data rate is 1250 Mbps.



Notes:
1. The parallel clock (tx_clkout or rx_clkout) is calculated as data rate/PCS-PMA interface width = 1250/10 = 125 MHz.
   When the Byte Serializer is set to Serialize x2 mode, tx_clkout and rx_clkout become 1250/20 = 62.5 MHz.
2. The serial clock is calculated as data rate/2. The PMA runs on a dual data rate clock.

In low latency datapath modes, the transmitter and receiver FIFOs are always enabled. Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks.

Related Information

Arria 10 Standard PCS Architecture on page 5-31

Word Aligner Manual Mode

1. Set the **RX word aligner mode** to **Manual (PLD controlled)**.
2. Set the **RX word aligner pattern length** option according to the PCS-PMA interface width.
3. Enter a hexadecimal value in the **RX word aligner pattern (hex)** field.

This mode adds `rx_patterndetect` and `rx_syncstatus`. You can select the **Enable rx_std_wa_patternalign port** option to enable `rx_std_wa_patternalign`. An active high on `rx_std_wa_patternalign` re-aligns the word aligner one time.

Note:
- `rx_patterndetect` is asserted whenever there is a pattern match.
- `rx_syncstatus` is asserted after the word aligner achieves synchronization.
- `rx_std_wa_patternalign` is also asserted to re-align and re-synchronize.
- If there is more than one channel in the design, `rx_patterndetect`, `rx_syncstatus` and `rx_std_wa_patternalign` become busses in which each bit corresponds to one channel.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-102: Manual Mode when the PCS-PMA Interface Width is 8 Bits

`tx_parallel_data` = 20'h3FC3BC and the word aligner pattern = 0x3BC

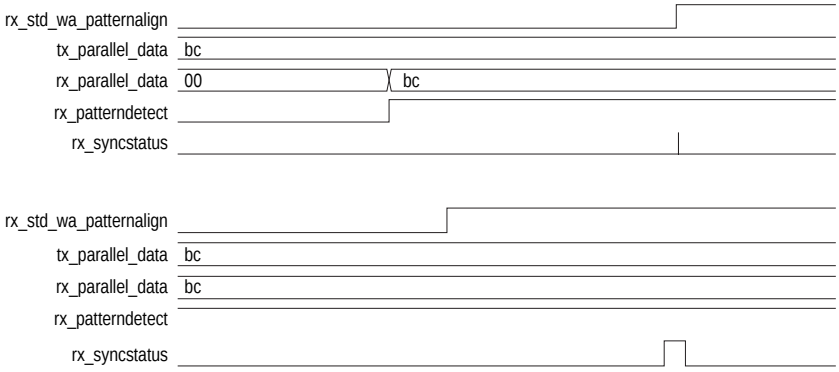


Figure 2-103: Manual Mode when the PCS-PMA Interface Width is 10 Bits

`tx_parallel_data` = 20'h3FC3BC and the word aligner pattern = 0x3BC

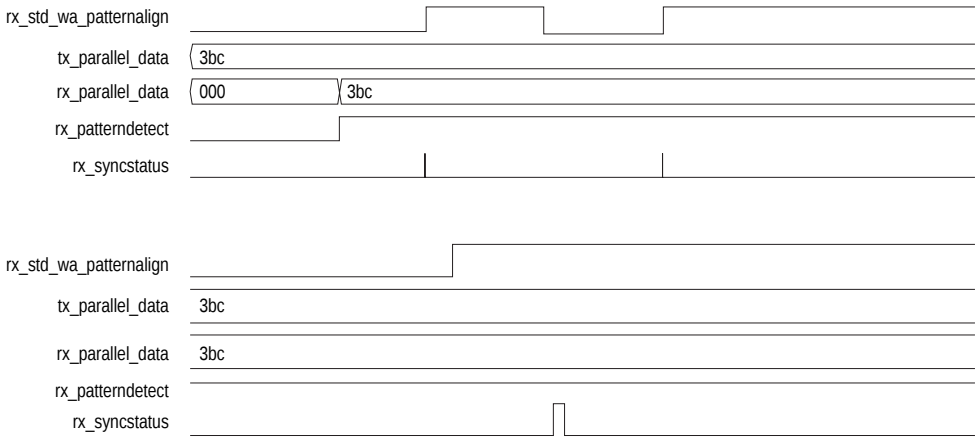


Figure 2-104: Manual Mode when the PCS-PMA Interface Width is 16 Bits

`tx_parallel_data` = 20'h3FC3BC and the word aligner pattern = 0x3BC

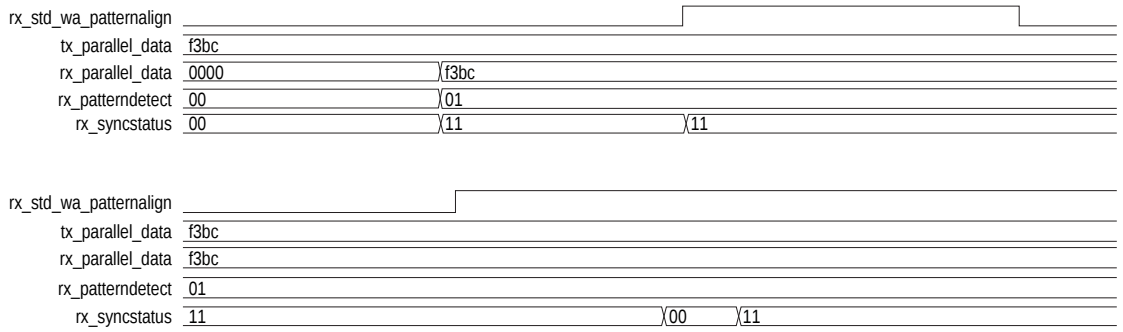
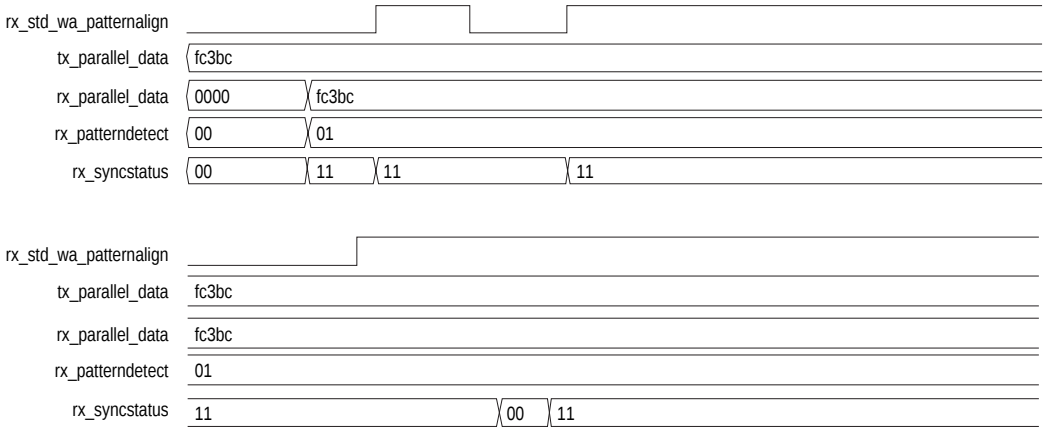


Figure 2-105: Manual Mode when the PCS-PMA Interface Width is 20 Bits

`tx_parallel_data` = 20'h3FC3BC and the word aligner pattern = 0x3BC



Word Aligner Synchronous State Machine Mode

To use this mode:

- Select the **Enable TX 8B/10B encoder** option.
- Select the **Enable RX 8B/10B decoder** option.

The 8B/10B encoder and decoder add the following additional ports:

- `tx_datak`
- `rx_datak`
- `rx_errdetect`
- `rx_disperr`
- `rx_runningdisp`

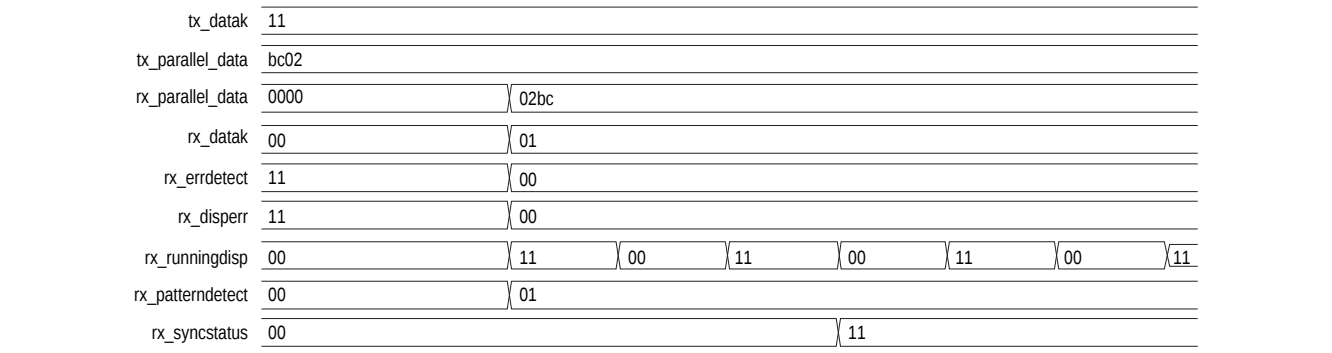
1. Set the **RX word aligner mode** to **synchronous state machine**.
2. Set the **RX word aligner pattern length** option according to the PCS-PMA interface width.
3. Enter a hexadecimal value in the **RX word aligner pattern (hex)** field.

The RX word aligner pattern is the 8B/10B encoded version of the data pattern. You can also specify the number of word alignment patterns to achieve synchronization, the number of invalid data words to lose synchronization, and the number of valid data words to decrement error count. This mode adds two additional ports: `rx_patterndetect` and `rx_syncstatus`.

Note:
- `rx_patterndetect` is asserted whenever there is a pattern match.
- `rx_syncstatus` is asserted after the word aligner achieves synchronization.
- `rx_std_wa_patternalign` is also asserted to re-align and re-synchronize.
- If there is more than one channel in the design, `tx_datak`, `rx_datak`, `rx_errdetect`, `rx_disperr`, `rx_runningdisp`, `rx_patterndetect`, and `rx_syncstatus` become busses in which each bit corresponds to one channel.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-106: Synchronization State Machine Mode when the PCS-PMA Interface Width is 20 Bits



RX Bit Slip

To use the RX bit slip, select **Enable rx_bitslip port**, and set the word aligner mode to **bit slip**. This adds `rx_bitslip` as an input control port. An active high edge on `rx_bitslip` slips one bit at a time, and when `rx_bitslip` is toggled, then the word aligner slips one bit at a time on every active high edge. You can verify this feature by monitoring `rx_parallel_data`.

The TX bit slip feature is optional and may or may not be enabled.

Figure 2-107: RX Bit Slip in 8-bit Mode

`tx_parallel_data` = 8'hbc

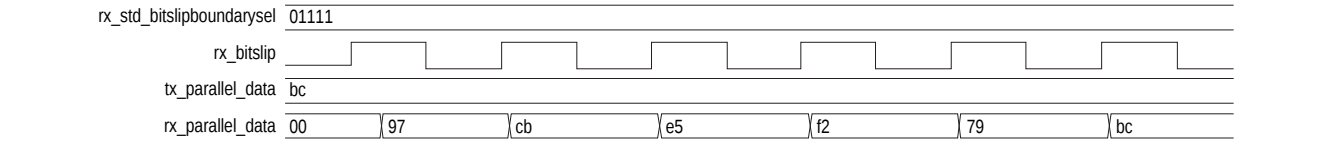


Figure 2-108: RX Bit Slip in 10-bit Mode

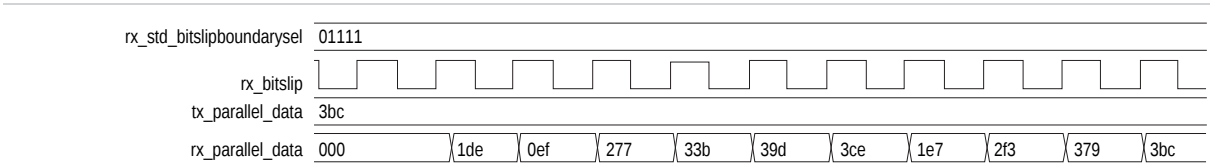


Figure 2-109: RX Bit Slip in 16-bit Mode

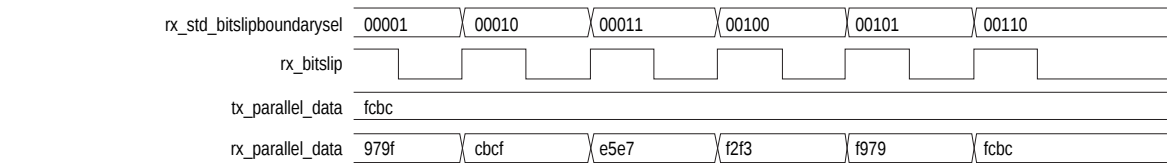
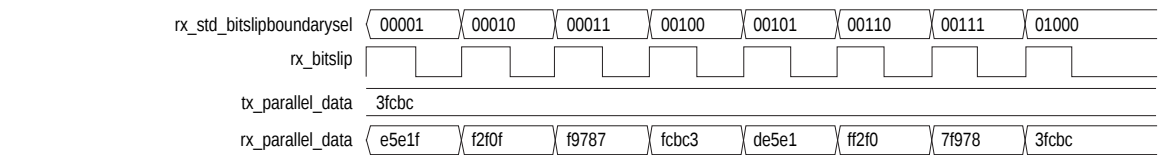


Figure 2-110: RX Bit Slip in 20-bit Mode



RX Polarity Inversion

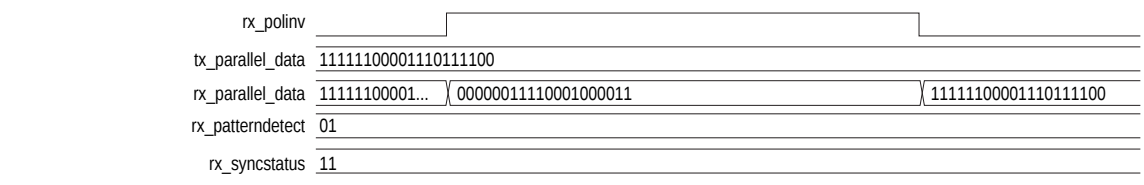
Receiver polarity inversion can be enabled in low latency, basic, and basic rate match modes; whereas, the word aligner is available in any mode.

To enable the RX polarity inversion feature, select the **Enable RX polarity inversion** and **Enable rx_polinv port** options.

This mode adds `rx_polinv`. If there is more than one channel in the design, `rx_polinv` is a bus in which each bit corresponds to a channel. As long as `rx_polinv` is asserted, the RX data received has a reverse polarity.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-111: RX Polarity Inversion



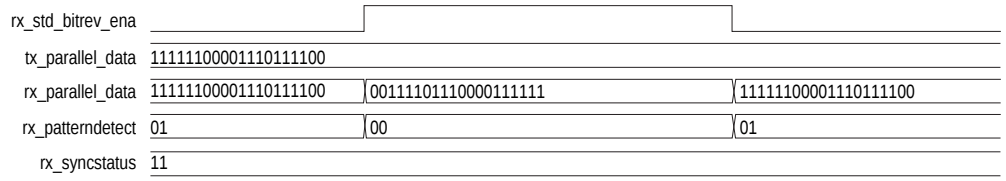
RX Bit Reversal

The RX bit reversal feature can be enabled in low latency, basic, and basic rate match mode. The word aligner is available in any mode, bit slip, manual, or synchronous state machine.

To enable this feature, select the **Enable RX bit reversal** and **Enable rx_std_bitrev_ena port** options. This adds `rx_std_bitrev_ena`. If there is more than one channel in the design, `rx_std_bitrev_ena` becomes a bus in which each bit corresponds to a channel. As long as `rx_std_bitrev_ena` is asserted, the RX data received by the core shows bit reversal.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-112: RX Bit Reversal



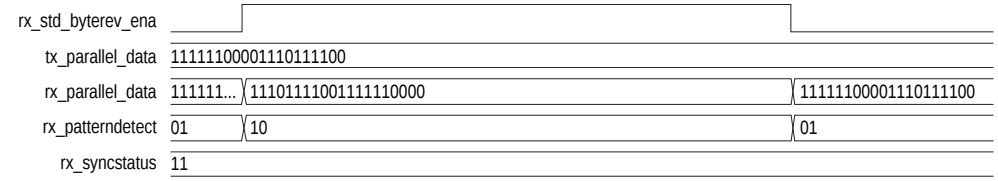
RX Byte Reversal

The RX byte reversal feature can be enabled in low latency, basic, and basic rate match mode; whereas, the word aligner is available in any mode.

To enable this feature, select the **Enable RX byte reversal** and **Enable rx_std_byterev_ena port** options. This adds `rx_std_byterev_ena`. If there is more than one channel in the design, `rx_std_byterev_ena` becomes a bus in which each bit corresponds to a channel. As long as `rx_std_byterev_ena` is asserted, the RX data received by the core shows byte reversal.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-113: RX Byte Reversal



Rate Match FIFO in Basic (Single Width) Mode

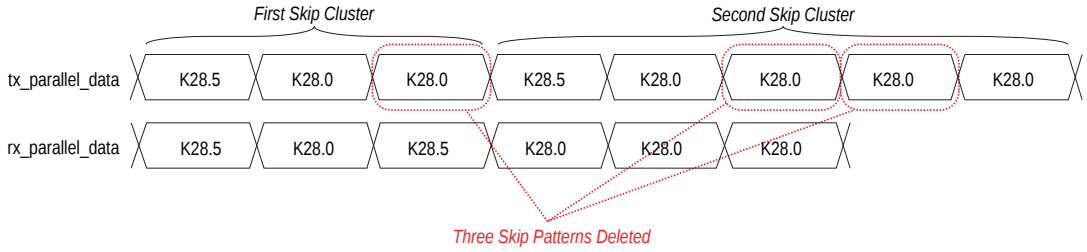
Only rate match FIFO operation is covered in these steps.

1. Select **basic (single width)** in the **RX rate match FIFO mode** list.
2. Enter values for the following parameters.

Parameter	Value	Description
RX rate match insert/delete +ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.

Parameter	Value	Description
RX rate match insert/delete -ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.

Figure 2-114: Rate Match FIFO Deletion with Three Skip Patterns Required for Deletion

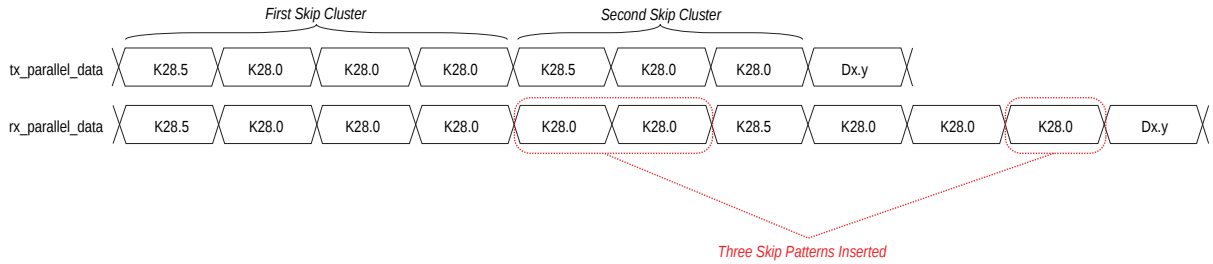


Note: /K28.5/ is the control patter and /K28.0/ is the skip pattern

In this example, the first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for a total of three skip patterns deletion requirement.

The rate match FIFO can insert a maximum of four skip patterns in a cluster, if there are no more than five skip patterns in the cluster after insertion.

Figure 2-115: Rate Match FIFO Deletion with Three Skip Patterns Required for Insertion



In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 2-116: Rate Match FIFO Becoming Full After Receiving D5

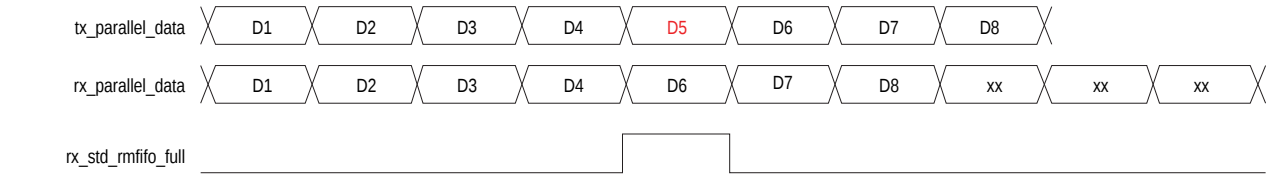
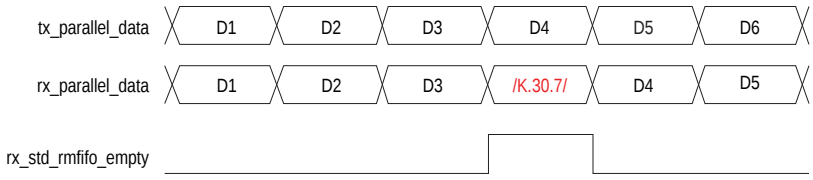


Figure 2-117: Rate Match FIFO Becoming Empty After Receiving D3



Rate Match FIFO Basic (Double Width) Mode

1. Select **basic (double width)** in the **RX rate match FIFO mode** list.
2. Enter values for the following parameters.

Parameter	Value	Description
RX rate match insert/delete +ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.
RX rate match insert/delete -ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.

The rate match FIFO can delete as many pairs of skip patterns from a cluster as necessary to avoid the rate match FIFO from overflowing. The rate match FIFO can delete a pair of skip patterns only if the two 10-bit skip patterns appear in the same clock cycle on the LSByte and MSByte of the 20-bit word. If the two skip patterns appear straddled on the MSByte of a clock cycle and the LSByte of the next clock cycle, the rate match FIFO cannot delete the pair of skip patterns.

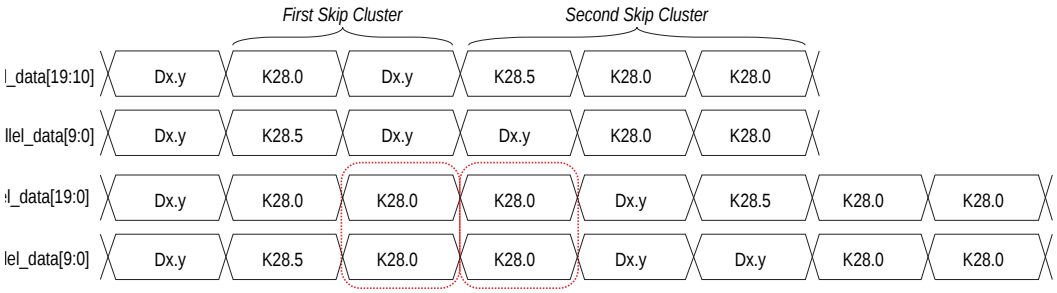
Figure 2-118: Rate Match FIFO Deletion with Four Skip Patterns Required for Deletion

/K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern.

In this example, the first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

The rate match FIFO can insert as many pairs of skip patterns into a cluster necessary to avoid the rate match FIFO from under running. The 10-bit skip pattern can appear on the MSByte, the LSByte, or both, of the 20-bit word.

Figure 2-119: Rate Match FIFO Deletion with Four Skip Patterns Required for Insertion



In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 2-120: Rate Match FIFO Becoming Full After Receiving the 20-Bit Word D5D6

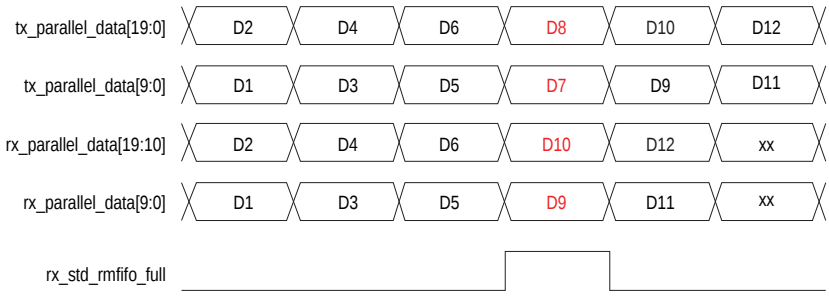
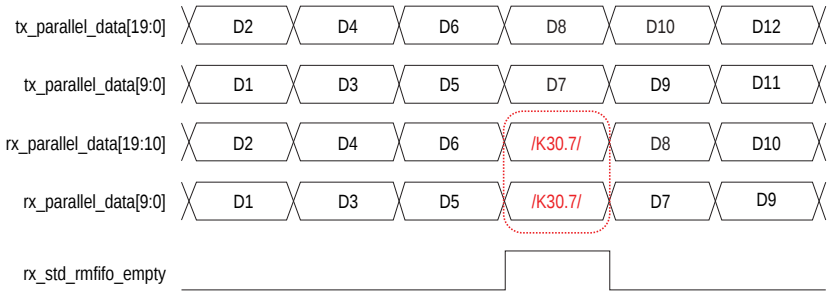


Figure 2-121: Rate Match FIFO Becoming Empty After Reading out the 20-Bit Word D5D6



How to Enable Low Latency in Basic

In the Arria 10 Transceiver Native PHY IP Core MegaWizard, use the following settings to enable low latency:

1. Select the **Enable 'Standard PCS' low latency mode** option.
2. Select either **low_latency** or **register FIFO** in the **TX FIFO mode** list.
3. Select either **low_latency** or **register FIFO** in the **RX FIFO mode** list.
4. Select either **Disabled** or **Serialize x2** in the **TX byte serializer mode** list.
5. Select either **Disabled** or **Serialize x2** in the **RX byte deserializer mode** list.
6. Ensure that **RX rate match FIFO mode** is **disabled**.
7. Set the **RX word aligner mode** to **bitslip**.
8. Set the **RX word aligner pattern length** to **7** or **16**.

Note: TX bitslip, RX bitslip, bit reversal, and polarity inversion modes are supported.

TX Bit Slip

To use the TX bit slip, select the **Enable TX bitslip** and **Enable tx_std_bitslipboundarysel port** options. This adds the `tx_std_bitslipboundarysel` input port. The TX PCS automatically slips the number of bits specified by `tx_std_bitslipboundarysel`. There is no port for TX bit slip. If there is more than one channel in the design, `tx_std_bitslipboundarysel` port becomes a bus in which each bit corresponds to one channel. You can verify this feature by monitoring the `rx_parallel_data` port.

The RX bit slip feature is optional and may or may not be enabled

Figure 2-122: TX Bit Slip in 8-bit Mode

`tx_parallel_data` = 8'hbc. `tx_std_bitslipboundarysel` = 5'b00001(bit slip by 1 bit).

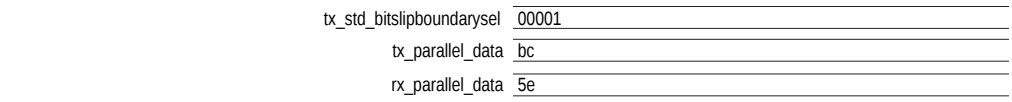


Figure 2-123: TX Bit Slip in 10-bit Mode

`tx_parallel_data` = 10'h3bc. `tx_std_bitslipboundarysel` = 5'b00011(bit slip by 3 bits).

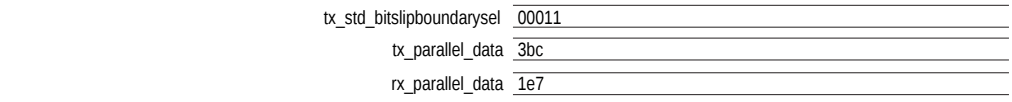


Figure 2-124: TX Bit Slip in 16-bit Mode

`tx_parallel_data` = 16'hfcbc. `tx_std_bitslipboundarysel` =5'b00011(bit slip by 3 bits).

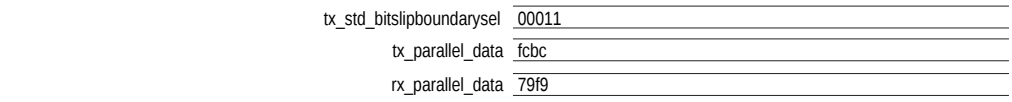
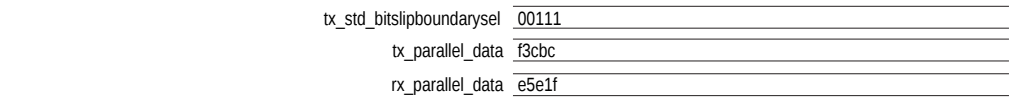


Figure 2-125: TX Bit Slip in 20-bit Mode

`tx_parallel_data` = 20'h3FCBC. `tx_std_bitslipboundarysel` = 5'b00111 (bit slip by 7 bits)



TX Polarity Inversion

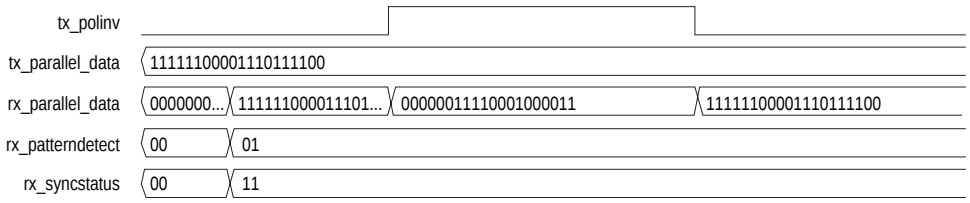
Transmitter polarity inversion can be enabled in low latency, basic, and basic rate match modes; whereas, the word aligner is available in any mode.

To enable the TX polarity inversion feature, select the **Enable TX polarity inversion** and **Enable tx_polinv port** options.

This mode adds `tx_polinv`. If there is more than one channel in the design, `tx_polinv` is a bus with each bit corresponding to a channel. As long as `tx_polinv` is asserted, the TX data transmitted has a reverse polarity.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-126: TX Polarity Inversion



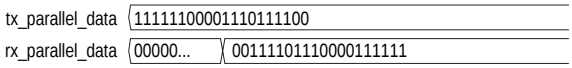
TX Bit Reversal

The TX bit reversal feature can be enabled in low latency, basic, and basic rate match mode; whereas, the word aligner is available in any mode. This feature is parameter-based, and creates no additional ports. If there is more than one channel in the design, all channels have TX bit reversal.

To enable this feature, select the **Enable TX bit reversal** option.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-127: TX Bit Reversal



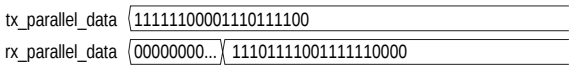
TX Byte Reversal

The TX byte reversal feature can be enabled in low latency, basic, and basic rate match mode; whereas, the word aligner is available in any mode. This feature is parameter-based, and creates no additional ports. If there is more than one channel in the design, all channels have TX byte reversal.

To enable this feature, select the **Enable TX byte reversal** option.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 2-128: TX Byte Reversal



How to Implement the Basic, Basic with Rate Match Transceiver Configuration Rules in Arria 10 Transceivers

Before you begin

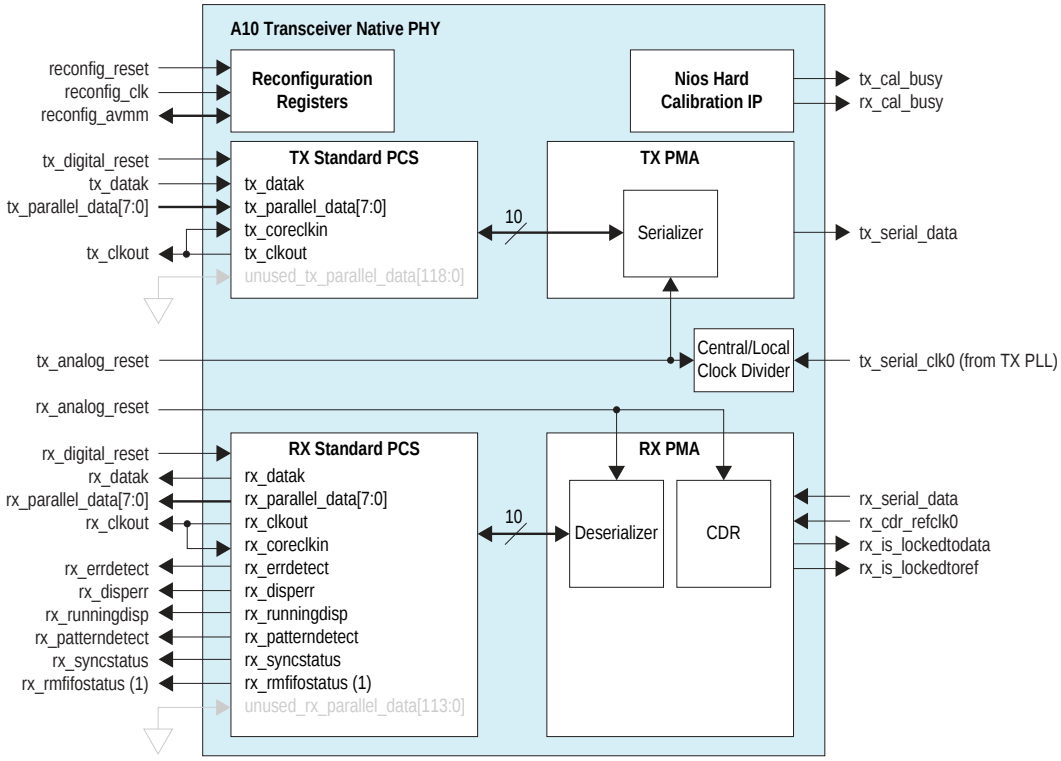
You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing your Basic protocol IP.

1. Open the MegaWizard Plug-In Manager and select the PHY IP.

Refer to **Select and Instantiate PHY IP** on page 2-2.

2. Select **Basic/Custom (Standard PCS)** or **Basic/Custom w/Rate Match (Standard PCS)** from the **Transceiver configuration rules** list located under **Datapath Options** depending on which configuration you want to use.
3. Use the parameter values in the tables in **Native PHY IP Parameter Settings for Basic, Basic with Rate Match Configurations** as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the setting to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

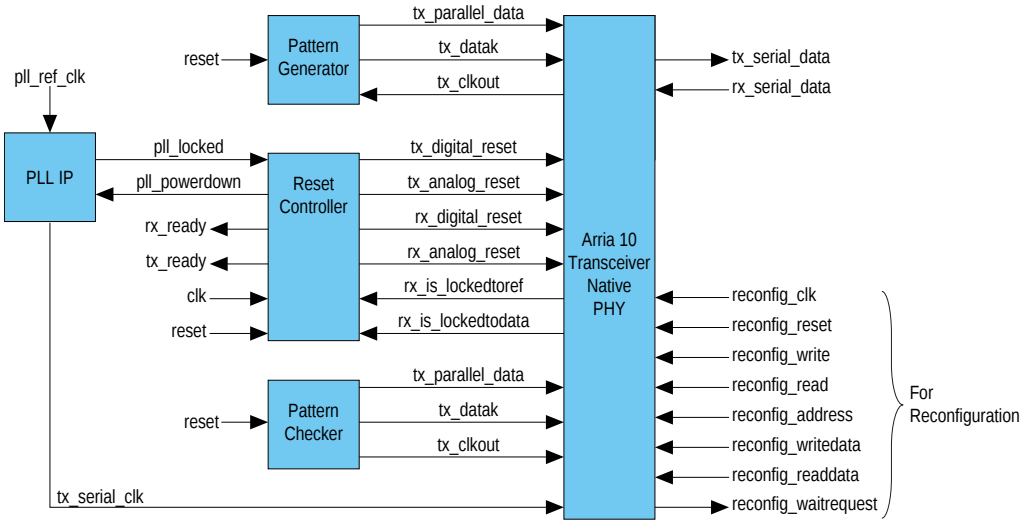
Figure 2-129: Signals and Ports of Native PHY IP for Basic, Basic with Rate Match Configurations



Note:
1. Only applies when using the Basic with Rate Match transceiver configuration rule.

5. Instantiate and configure your PLL.
6. Create a transceiver reset controller.
7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in **Native PHY IP Parameter Settings for Basic, Basic with Rate Match Configurations** to connect the ports.

Figure 2-130: Connection Guidelines for a Basic/Custom Design



8. Simulate your design to verify its functionality.

Related Information

- **Arria 10 Standard PCS Architecture** on page 5-31
  For more information about Standard PCS architecture
- **Arria 10 PMA Architecture** on page 5-1
  For more information about PMA architecture
- **Using PLLs and Clock Networks** on page 3-40
  For more information about implementing PLLs and clocks
- **PLLs** on page 3-3
  PLL architecture and implementation details
- **Resetting Transceiver Channels** on page 4-1
  Reset controller general information and implementation details
- **Standard PCS and PMA Ports** on page 2-52
  Port definitions for the Transceiver Native PHY Standard Datapath

Native PHY IP Parameter Settings for Basic, Basic with Rate Match Configurations

Table 2-147: General and Datapath Options Parameters

Parameter	Range
Device speed grade	**fastest**
Message level for rule violations	• **error** • **warning**

Parameter	Range
Transceiver configuration rules	• Basic/Custom (Standard PCS) • Basic/Custom w/Rate Match (Standard PCS)
Transceiver mode	• TX/RX Duplex • TX Simplex • RX Simplex
Number of data channels	1 to 96
Data rate	125 Mbps to 10 Gbps
Enable reconfiguration between Standard and Enhanced PCS datapaths	On/Off
Enable simplified data interface	On/Off

Table 2-148: TX PMA Parameters

Parameter	Range
TX channel bonding mode	• Not bonded • PMA bonding • PMA/PCS bonding
PCS TX channel bonding master	Auto, 0, 1
Actual PCS TX channel bonding master	0, 1
TX local clock division factor	1, 2, 4, 8
Number of TX PLL clock inputs per channel	1, 2, 3, 4
Initial TX PLL clock input selection	0
Enable tx_pma_clkout port	On/Off
Enable tx_pma_div_clkout port	On/Off
tx_pma_div_clkout division factor	Disabled, 1, 2, 33, 40, 66
Enable tx_pma_elecidle port	On/Off
Enable tx_pma_qpipullup port (QPI)	On/Off
Enable tx_pma_qpipulldn port (QPI)	On/Off
Enable tx_pma_txdetectrx port (QPI)	On/Off
Enable tx_pma_rxfound port (QPI)	On/Off

Table 2-149: RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1, 2, 3, 4, 5**
Selected CDR reference clock	**0, 1, 2, 3, 4**
Selected CDR reference clock frequency	Legal range defined by Quartus II
PPM detector threshold	**62.5, 100, 125, 200, 250, 300, 500, 1000**
Decision feedback equalization mode	• **Disabled** • **Fixed tap** • **Floating tap**
Enable rx_pma_clkout port	**On/Off**
Enable rx_pma_div_clkout port	**On/Off**
rx_pma_div_clkout division factor	**Disabled, 1, 2, 33, 40, 50, 66**
Enable rx_pma_clkslip port	**On/Off**
Enable rx_pma_qpipulldn port (QPI)	**On/Off**
Enable rx_is_lockedtodata port	**On/Off**
Enable rx_is_lockedtoref port	**On/Off**
Enable rx_set_locktodata and rx_set_locktoref ports	**On/Off**
Enable rx_seriallpbken port	**On/Off**
Enable PRBS verifier control and status ports	**On/Off**

Table 2-150: Standard PCS Parameters

Parameter	Range
Standard PCS / PMA interface width	**8, 10, 16, 20**
FPGA Fabric / Standard TX PCS interface width	**8, 10, 16, 20, 32, 40**
FPGA Fabric / Standard RX PCS interface width	**8, 10, 16, 20, 32, 40**
Enable Standard PCS low latency mode	• **On/Off** • **Off** (for Basic with Rate Match)
TX FIFO mode	• **low_latency** • **register_fifo**
RX FIFO Mode	• **low_latency** • **register_fifo**
Enable tx_std_pcfifo_full port	**On/Off**
Enable tx_std_pcfifo_empty port	**On/Off**

Parameter	Range
Enable rx_std_pcfifo_full port	**On/Off**
Enable rx_std_pcfifo_empty port	**On/Off**
TX byte serializer mode	• **Disabled** • **Serialize x2**
RX byte deserializer mode	• **Disabled** • **Deserialize x2**
Enable TX 8B/10B encoder	**On/Off**
Enable TX 8B/10B disparity control	**On/Off**
Enable RX 8B/10B decoder	**On/Off**
RX rate match FIFO mode	• **Disabled** • **Basic (single width)** (for Basic with Rate Match) • **Basic (double width)** (for Basic with Rate Match)
RX rate match insert / delete -ve pattern (hex)	User-defined value
RX rate match insert / delete +ve pattern (hex)	User-defined value
Enable rx_std_rmfifo_full port	**On/Off**
Enable rx_std_rmfifo_empty port	**On/Off**
Enable TX bit slip	**On/Off**
Enable tx_std_bitslipboundarysel port	**On/Off**
RX word aligner mode	• **bitslip** • **manual (PLD controlled)** • **synchronous state machine**
RX word aligner pattern length	**7**, **8**, **10**, **16**, **20**, **32**, **40**
RX word aligner pattern (hex)	User-defined value
Number of word alignment patterns to achieve sync	0-255
Number of invalid data words to lose sync	0-63
Number of valid data words to decrement error count	0-255
Enable rx_std_wa_patternalign port	**On/Off**
Enable rx_std_wa_a1a2size port	**On/Off**
Enable rx_std_bitslipboundarysel port	**On/Off**
Enable rx_bitslip port	**On/Off**
Enable TX bit reversal	**On/Off**

Parameter	Range
Enable TX byte reversal	On/Off
Enable TX polarity inversion	On/Off
Enable tx_polinv port	On/Off
Enable RX bit reversal	On/Off
Enable rx_std_bitrev_ena port	On/Off
Enable RX byte reversal	On/Off
Enable rx_std_byterev_ena port	On/Off
Enable RX polarity inversion	On/Off
Enable rx_polinv port	On/Off
Enable rx_std_signaldetect port	On/Off
Enable PCIe dynamic datarate switch ports	Off
Enable PCIe pipe_hclk_in and pipe_hclk_out ports	Off
Enable PCIe Gen3 analog control ports	Off
Enable PCIe electrical idle control and status ports	Off
Enable PCIe pipe_rx_polarity port	Off

Table 2-151: Dynamic Reconfiguration Parameters

Parameter	Range
Enable dynamic reconfiguration	On/Off
Share reconfiguration interface	On/Off
Enable embedded JTAG AVMM master	On/Off

Table 2-152: Generation Options Parameters

Parameter	Range
Generate parameter documentation file	On/Off

Design Considerations for Data Rates Above 17.4 Gbps Using Arria 10 GT Channels

This section provides information on using the Arria 10 GT transceiver channels to achieve data rates from 17.4 to 28.1 Gbps. Arria 10 GT transceiver channels are used to implement data rates above 17.4 Gbps.

GT channels can be used in Enhanced PCS Low Latency mode to support data rates from 17.4 Gbps to 28.1 Gbps. GT channels can also operate in PCS-Direct configuration for data rates up to 28.1 Gbps. When GT channels are used in PCS-Direct configuration, the PCS blocks are bypassed. The serializer / deserializer in GT channels supports 64 bit and 128 bit serialization factors.

Arria 10 GT Channel Usage

All Arria 10 GT devices have a total of 16 GT transceiver channels that can support data rates up to 28.1 Gbps.

In Arria 10 GT devices, each transceiver bank supports up to 4 GT channels that can operate as a duplex channel, TX only, or RX only channel. Transceiver banks GXBL1E, GXBL1F, GXBL1G, and GXBL1H each contain four GT transceiver channels - 0, 1, 3 and 4. Channels 2 and 5 can only be configured as GX transceiver channels.

Table 2-153: Valid Permutations for GT and GX Channel Configuration in Transceiver Banks GXBL1E, GXBL1F, GXBL1G, and GXBL1H for Channels 0, 1, and 2

GT Transceiver Channel	Configuration A	Configuration B	Configuration C	Configuration D	Configuration E	Configuration F
Ch2	Unusable	Unusable	Unusable	GX	GX	GX
Ch1	GT	GT	GX	Unusable	GT	GX
Ch0	GT	GX	GT	GT	Unusable	GX

Notes:

- If both the channels 0 and 1 are used with either one of them configured as a GT channel and the other one as GT or GX, then channel 2 is unusable. For all other configurations of channels 0 and 1, channel 2 can only be configured as a GX channel.
- If either channel 0 or 1 is used as a GT channel, then the ATX PLL adjacent to channel 0 and 1 needs to be reserved for GT channel configurations.

Table 2-154: Valid Permutations for GT and GX Channel Configuration in Transceiver Banks GXBL1E, GXBL1F, GXBL1G, and GXBL1H for Channels 3, 4, and 5

GT Transceiver Channel	Configuration A	Configuration B	Configuration C	Configuration D	Configuration E	Configuration F
Ch5	Unusable	Unusable	Unusable	GX	GX	GX
Ch4	GT	GT	GX	Unusable	GT	GX
Ch3	GT	GX	GT	GT	Unusable	GX

Notes:

- If both the channels 3 and 4 are used with either one of them configured as a GT channel and the other one as GT or GX, then channel 5 is unusable. For all other configurations of channels 3 and 4, channel 5 can only be configured as a GX channel.
- If either channel 3 or 4 is used as a GT channel, then the ATX PLL adjacent to channel 3 and 4 needs to be reserved for GT channel configurations.

Transceiver PHY IP

Arria 10 GT transceiver channels are implemented using the Native PHY IP with the Basic (Enhanced PCS) transceiver configuration rule.

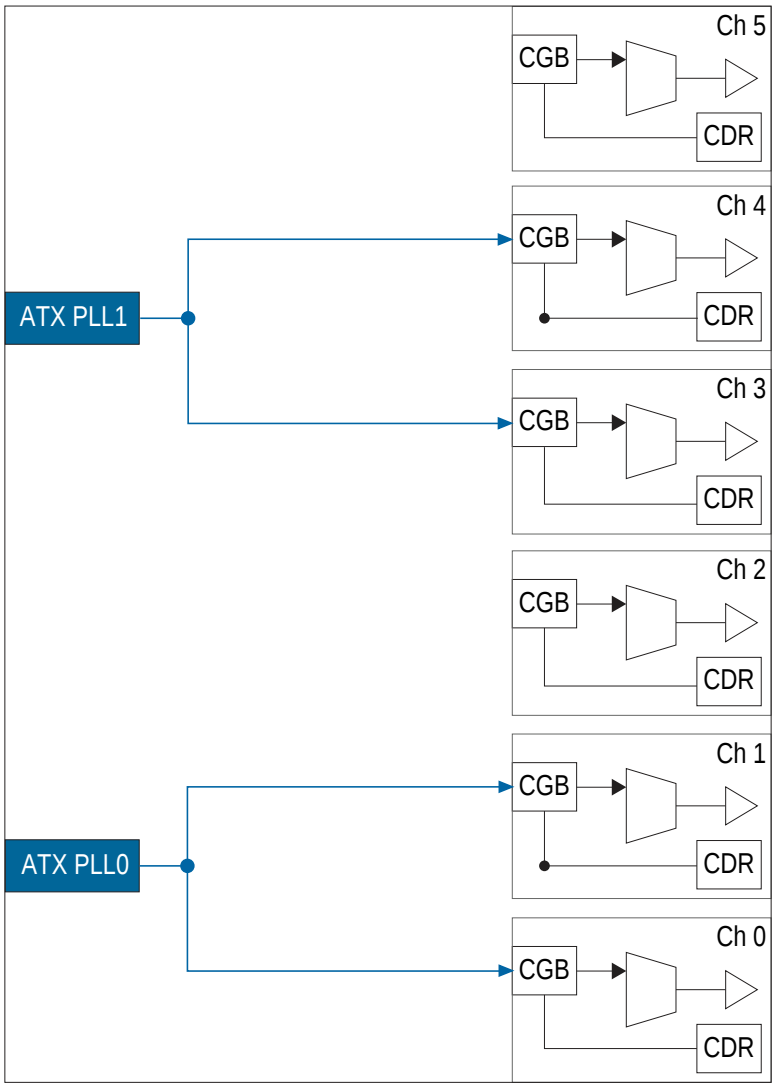
- To support data rates from 17.4 Gbps to 28.1 Gbps, the Enhanced PCS must be configured in low latency mode. To configure the Enhanced PCS in low latency mode, do not enable any functional blocks in the Enhanced PCS (i.e. disable Block Sync, Gearbox, Scrambler & Encoder).
- You can also use the PCS-Direct mode, for data rates from 17.4 Gbps to 28.1 Gbps. PCS-Direct will be available in a future release of the QuartusII software.

You can bundle several GT transceiver channels with one Native PHY IP instantiation, but you will need to instantiate a separate ATX PLL IP for every ATX PLL used.

PLL and GT Transceiver Channel Clock Lines

The ATX PLL is used to provide the clock source for the GT transceiver channels. Each ATX PLL has two dedicated GT clock lines which connect the PLL directly to the GT transceiver channels within a transceiver bank. The top ATX PLL drives channels 3 and 4, and the bottom ATX PLL drives channels 0 and 1. These connections bypass the rest of the clock network for higher performance.

Figure 2-131: GT Channel Configuration



When both the channels 0 and 1 are configured as GT channels, they are driven by the same ATX PLL and have to be configured to run at the same data rates. This is also true for channels 3 and 4 when they are configured as GT channels.

Skew is expected between GT channels and the exact values are pending device characterization. Currently, GT channel bonding is not supported.

Reset Controller

Each GT channel instantiated will have independent analog and digital reset ports. Refer to the *Resetting Transceiver Channels* chapter for more details on designing a reset controller to reset these ports.

Related Information

Resetting Transceiver Channels on page 4-1
Reset controller general information and implementation details

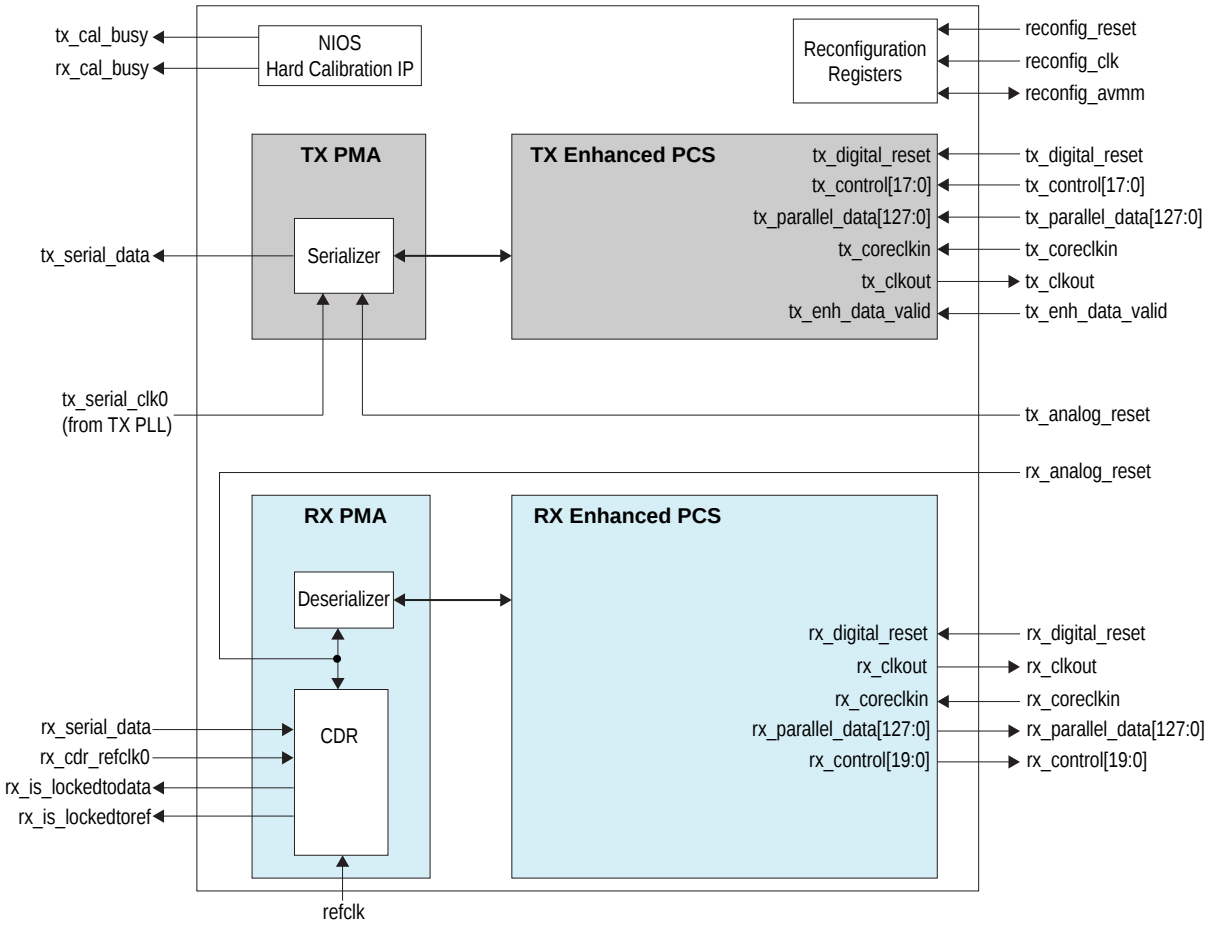
How to Implement Designs for Data Rates Above 17.4 Gbps Using Enhanced PCS in Low Latency Mode

Before you begin

You should be familiar with the Enhanced PCS and PMA architecture, PLL architecture, and the reset controller.

1. Open the MegaWizard Plug-In Manager and select the Native PHY IP. Refer to **Select and Instantiate PHY IP** on page 2-2 for detailed steps.
2. Select **Basic (Enhanced PCS)** from the **Transceiver configuration rules** list located under **Datapath Options**.
3. Use the parameter values in the tables in **Transceiver Native PHY IP Parameters for Basic (Enhanced PCS) Transceiver Configuration Rule** for each input of the Arria 10 Transceiver Native PHY MegaWizard as a starting point. Or, you can use the protocol presets described in **Preset Configuration Options**. You can then modify the settings to meet your specific requirements.
   - Ensure that the data rate is between 17400 and 28100 Mbps. Select a CDR reference clock to match your data rate.
   - Set the Enhanced PCS / PMA interface width to 64 bits.
   - Set the FPGA Fabric / Enhanced PCS interface width must be 64 bits.
   - You can enable RX/TX FIFO double width mode to create a FPGA fabric / PCS interface width of 128 bits.
   - Click **Finish** to generate the Native PHY IP (this is your RTL file).

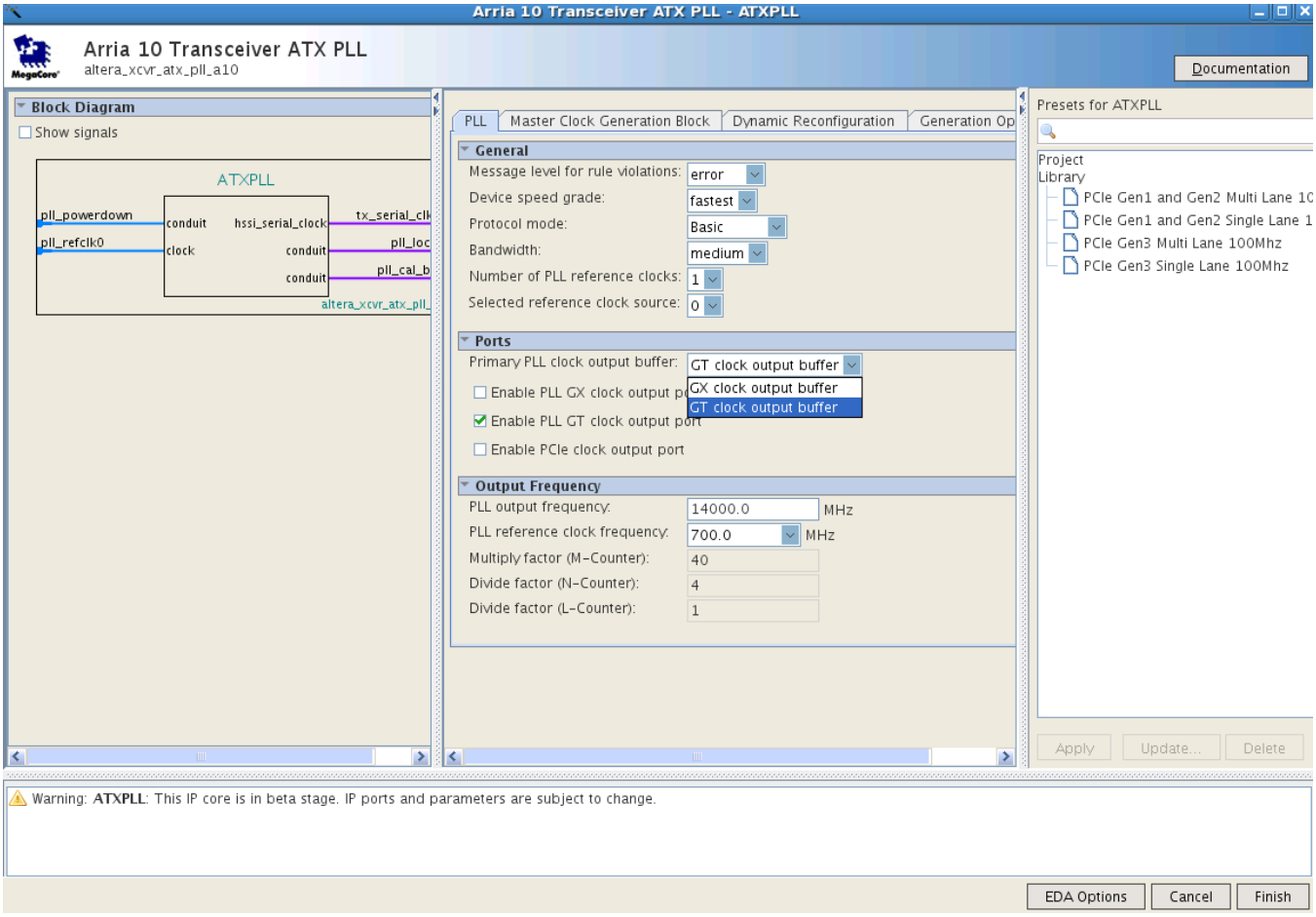
Figure 2-132: Signals and Ports of the Native PHY for Basic (Enhanced PCS) Transceiver Configuration Rule for Data Rates Above 17.4 Gbps and FPGA Fabric / PCS Interface width of 128 bits



Note: For GT transceiver channels, all the individual functional blocks within the enhanced PCS are not enabled (and are bypassed) to provide the lowest latency PCS from the PMA. `rx_control` and `tx_control` ports are not used.

4. Open the **MegaWizard Plug-In Manager** and select ATX PLL IP. Refer to **Instantiating ATX PLL** on page 3-5 for detailed steps.
5. Configure the ATX PLL IP using the MegaWizard Plug-In Manager.
   - Select the GT clock output buffer.
   - Enable the PLL GT clock output port.
   - Set the PLL output clock frequency to the Native PHY IP recommended frequency.

Figure 2-133: ATX PLL IP with GT Clock Lines Enabled



6. Create a transceiver reset controller. Refer to **Resetting Transceiver Channels** on page 4-1 for more details about configuring the reset IP.
7. Connect the Native PHY IP to the PLL IP and the reset controller.
   - The ATX PLL's port `tx_serial_clk_gt` represents the dedicated GT clock lines. Connect this port to the Native PHY IP's `tx_serial_clk0` port. Quartus II software will automatically use the dedicated GT clocks instead of the x1 clock network.

Note: The Quartus II software version 13.1 does not perform legality checks for PCS to FPGA Fabric speeds. The transceiver data rate is checked, but the PCS to FPGA fabric data rate can be set too high. For example, if the PCS / PMA interface width is less than 40 bits for data rates above 17.4Gbps, then the PCS to FPGA fabric data rate can be set too high. PCS interface legality checking will be available in a future release of the Quartus-II software.

Simulating the Transceiver Native PHY IP Core

Use simulation to verify the Native PHY transceiver functionality. The Quartus II software supports register transfer level (RTL) and gate-level simulation in both ModelSim-Altera and third-party simulators. You run simulations using your Quartus II project files.

The following simulation flows are available:

- NativeLink with ModelSim-Altera: This flow simplifies simulation by allowing you to start a simulation from the Quartus II software. This flow automatically creates a simulation script and compiles design files, IP simulation model files, and Altera simulation library models.
- Custom Flow: This flow allows you to customize simulation for more complex requirements. You can use this flow to compile design files, IP simulation model files, and Altera simulation library models manually.

You can simulate the following three netlists:

- The RTL functional netlist: This netlist provides cycle-accurate simulation using Verilog HDL, SystemVerilog, and VHDL design source code. Altera and third-party EDA vendors provide the simulation models.
- The post-synthesis gate-level functional netlist: This netlist verifies functionality after synthesis.
- The gate-level timing netlist: This netlist verifies both functionality and timing.

Prerequisites to Simulation

Before you can simulate your design, you must have successfully passed Quartus II Analysis and Synthesis.

Related Information

Simulating Altera Designs

NativeLink Simulation Flow

The NativeLink settings available in the Quartus II software allow you to specify your simulation environment, simulation scripts, and testbenches. The Quartus II software saves these settings in you project. Once you specify the NativeLink settings, you can start simulations easily from the Quartus II software.

How to Use NativeLink to Specify a ModelSim-Altera Simulation

Complete the following steps to specify the directory path and testbench settings for your simulator:

1. On the **Tools** menu, click **Options**, and then click **EDA Tool Options**.
2. Browse to the directory for your simulator. The following table lists the directories for supported simulators:

Table 2-155: Simulator Path

Simulator	Path
Mentor Graphics ModelSim-Altera	*<drive>*:*<simulator install path>***win32aloem** (Windows) /*<simulator install path>*/**bin** (Linux)

3. On Assignments menu, click **Settings**.
4. In the **Category** list, under **EDA Tool Settings** select **Simulation**.
5. In the **Tool name** list, select your simulator.

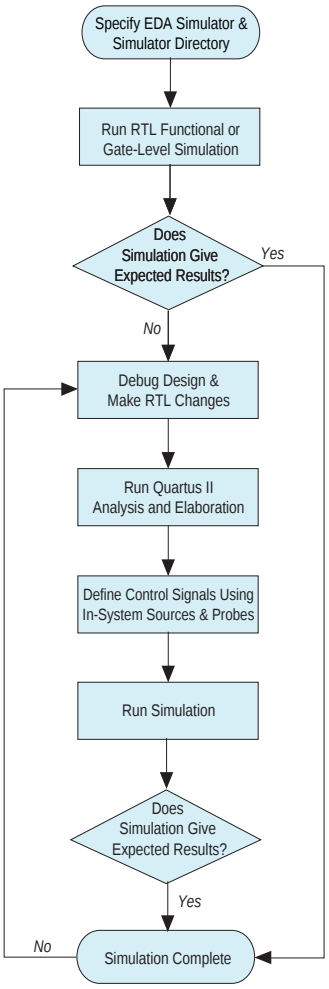
   Note: ModelSim refers to ModelSim SE, PE, and DE. These simulators use the same commands as QuestaSim. ModelSim-Altera refers to ModelSim-Altera Starter Edition and ModelSim-Altera Subscription Edition.

6. In the **Output directory:**, browse to the directory for your output files.
7. To map illegal HDL characters, turn on **Map illegal HDL characters**.
8. To filter netlist glitches , turn on **Enable glitch filtering**. This option is not available for Arria 10 devices in the Beta2 release.
9. Complete the following steps to specify additional options for NativeLink automation:
   a. Turn on **Compile testbench**.
   b. Click **Test Benches**.
      The **Test Benches** dialog box appears.
   c. Click **New**.
   d. Under **Create new test bench settings**, for **Test bench name** type the testbench name. For Top level module in the testbench, type the top-level module name. These names should match the actual testbench module names.
   e. Under the **Simulation period**, turn on **Run simulation until all vector stimiuli are used**.
   f. Click **OK**.

How to Use NativeLink to Run a ModelSim-Altera RTL Simulation

The following figure illustrates the high-level steps for NativeLink with ModelSim-Altera simulation.

Figure 2-134: NativeLink Simulation Flow Diagram



Complete the following steps to run an RTL functional simulation:

1. Open your Quartus II project.
2. On the Tools menu, select **Run Simulation Tool**, then select **RTL Simulation** or **Gate Level Simulation**.
3. Complete the following steps to define control signals necessary for simulation.
4. On the Tools menu, select **In-System Sources and Probes Editor**.

   Note: To re-simulate after correcting errors, you must first rerun Quartus II Analysis and Elaboration and re-instantiate control signals that you defined using the In-System Sources and Probe Editor. The In-System Sources and Probe Editor can only access the pins of the device. Consequently, you must route any signal that you want to observe to the top-level of your design.

5. To monitor additional signals, highlight the desired instances or nodes in **Instance**, and right-click **Add wave**.
6. Select **Simulate** and then **Run**.
7. Specify the simulation duration.
8. Complete the following steps to restart the simulation:
   a. On the ModelSim-Altera Simulate menu, select **restart**, then click **ok**.

This action clears the existing waves.

b. Highlight **run** and select the appropriate options to run the simulation.

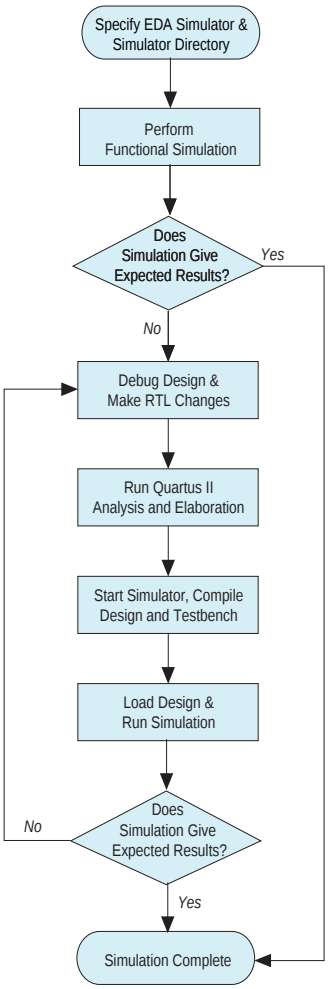
Related Information

Simulating the Transceiver Native PHY IP Core on page 2-256

How to Use NativeLink to Specify Third-Party RTL Simulators

The following figure illustrates the high-level steps for NativeLink with Third-Party EDA RTL Simulations.

Figure 2-135: Using NativeLink with Third-Party Simulators



Complete the following steps to specify the directory path and testbench settings for your simulator:

1. On the **Tools** menu, click **Options**, and then click **EDA Tool Options**.
2. Browse to the directory for your simulator. The following table lists the directories for supported third-party simulators:

Table 2-156: Simulator Path

Simulator	Path
Mentor Graphics ModelSim Mentor Graphics QuestaSim	<*drive*>:\<*simulator install path*>**win32** (Windows) /<*simulator install path*>/**bin** (Linux)
Synopsys VCS/VCS MX	/<*simulator install path*>/**bin** (Linux)
Cadence Incisive Enterprise	/<*simulator install path*>/**tools**/**bin** (Linux)
Aldec Active-HDL Aldec Riviera-Pro	<*drive*>:\<*simulator install path*>**bin** (Windows) /<*simulator install path*>/**bin** (Linux)

3. On Assignments menu, click **Settings**.
4. In the **Category** list, under **EDA Tool Settings**, select **Simulation**.
5. In the **Tool name** list, select your simulator.
6. To enable your simulator, on the **Tools** menu, click **Options** and then click **License Setup** . Make necessary changes for EDA tool licenses.
7. Compile your design and testbench files.
8. Load the design and run the simulation in the EDA tool.

To learn more about third-party simulators, click on the appropriate link below.

Related Information

- **Mentor Graphics ModelSim and QuestaSim Support**
- **Synopsys VCS and VCS MX Support**
- **Cadence Incisive Enterprise Simulator Support**
- **Aldec Active-HDL and Riviera-Pro Support**

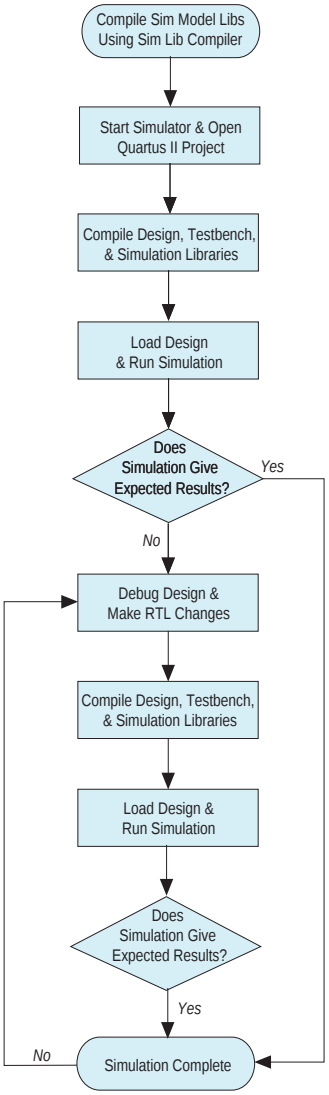
Custom Simulation Flow

The custom simulation flow allows you to customize the simulation process for more complex simulation requirements. This flow allows you to control the following aspects of your design:

- Component binding
- Compilation order
- Run commands
- IP cores
- Simulation library model files

The following figure illustrates the steps for custom flow simulation. If you use a simulation script, you can automate some of the steps.

Figure 2-136: Custom flow Simulation


Compile Sim Model Libs Using Sim Lib Compiler
Start Simulator & Open Quartus II Project
Compile Design, Testbench, & Simulation Libraries
Load Design & Run Simulation
Does Simulation Give Expected Results?
Yes
No
Debug Design & Make RTL Changes
Compile Design, Testbench, & Simulation Libraries
Load Design & Run Simulation
Does Simulation Give Expected Results?
Yes
No
Simulation Complete

How to Use the Simulation Library Compiler

The Simulation Library Compiler compiles Altera simulation libraries for supported simulation tools. The Simulation Library Compiler saves simulation files in the output directory you specify.

Note: Because the ModelSim-Altera software provides precompiled simulation libraries, you do not have to compile simulation libraries if you are use the ModelSim-Altera software.

Complete the following steps to compile the simulation model libraries using the Simulation Library Compiler:

1. On the Tools menu, click **Launch Simulation Library Compiler**.
2. Under **EDA simulation tool**, for the **Tool name**, select your simulation tool.
3. Under **Executable location**, browse to the location of the simulation tool you specified. You must specify this location before you can run the EDA Simulation Library Compiler.
4. Under **Library families**, select one or more family names and move them to the **Selected families** list.

5. Under **Library language**, select **Verilog**, **VHDL**, or both.
6. In the **Output directory** field, specify a location in which to store the compiled libraries.
7. Click **Start Compilation**.

Complete the following steps to add the simulation files to your project:

1. On the Assignments menu, click **Settings**.
2. In the **Category** list, select **Files**.
3. Click **Browse** to open the **Select File** dialog box and select one or more files in the **Files** list to add to your project.
4. Click **Open**, and then **Add** to add the selected file(s) to your project.
5. Click **OK** to close the **Settings** dialog box.

Related Information

- **Preparing for EDA Simulation**
- **Altera Simulation Models**

How to Generate Scripts

When you compile your design, the Quartus II software and Qsys automatically generate simulation scripts for the supported third-party simulation tools. You can also use the `ip-make-simscript` utility to generate scripts for multiple IP cores or a Qsys system.

Custom Simulation Scripts

You can automate simulations by creating customized scripts. You can generate scripts manually. In addition, you can use NativeLink to generate a simulation script as a template and then make the necessary changes. The following table shows a list of script directories NativeLink generates.

Table 2-157: NativeLink Generated Scripts for Third-Party RTL Simulation

Simulator	Simulation File	Use
Mentor Graphics ModelSim or QuestaSim	**/simulation/ modelsim/modelsim_setup.do**	Source directly with your simulator.
Aldec Riviera Pro	**/simulation/ aldec/rivierapro_setup.tcl**	Source directly with your simulator
Synopsys VCS	**/simulation/synopsys/vcs/vcs_setup.sh**	Add your testbench file name to this options file to pass the file to VCS using the –file option. If you specify a testbench file for NativeLink and do not choose to simulate, NativeLink generates an script that runs VCS.

Simulator	Simulation File	Use
Synopsys VCS MX	**/simulation/synopsys/vcsmx/vcsmx_setup.sh**	Run this script at the command line using `quartus_sh-t <script>`. Any testbench you specify with NativeLink is included in this script.
Cadence Incisive (NCSim)	**/simulation/cadence/ncsim_setup.sh**	Run this script at the command line using `quartus_sh -t` <*script*>. Any testbench you specify with NativeLink is included in this script.

Qsys Simulation Scripts

If you use Qsys to create your design, you have the following two options to create scripts:

Generate Scripts in Qsys

Qsys system generation creates the interconnect between components. It also generates files for synthesis and simulation, including the **.spd** files necessary for the `ip-make-simscript` utility. Complete the following steps to generate a simulation module for a Qsys system.

1. On the Generate menu, click **Generate**.
   The Generation dialog box appears.
2. Under **Create simulation model**, select your HDL.
   Note that Qsys appends **simulation** to your project directory to create a subdirectory for simulation files.
3. Click **Generate**. Qsys generates simulation files and scripts for the Aldec Riviera Pro, Cadence NCSim, and Mentor Graphics ModelSim simulation tools.
4. You can run the scripts to compile the required device libraries and system design files in the correct order. Then, elaborate or load the top-level design for simulation.

Use the ip-make-simscript Utility

This utility generates simulation command scripts for multiple IP cores or Qsys systems. To use this command, you must specify Simulation Package Descriptor (**.spd**) files for each IP core or Qsys system. This utility compiles IP simulation models into simulation libraries. Complete the following steps to use this command in Qsys:

1. On the Qsys Tools menu, select **Nios II Command Shell [gcc4]**.
   A command shell appears.
2. To get usage information for the `ip-make-simscript` utility, type the following command:
   `ip-make-simscript --help`
3. Then, type `ip-make-simscript` with the appropriate arguments.
4. You can run the scripts to compile the required device libraries and system design files in the correct order. Then, elaborate or load the top-level design for simulation.

Related Information

- **Simulating Altera Designs**
- **Creating a System with Qsys**

PLLs and Clock Networks

3

2013.12.02

UG-A10XCVR

Subscribe Send Feedback

This chapter provides information about the Arria 10 transceiver clocking architecture. This chapter describes the transceiver phase locked loops (PLLs), internal clocking architecture and the clocking options for the transceiver and the FPGA fabric interface.

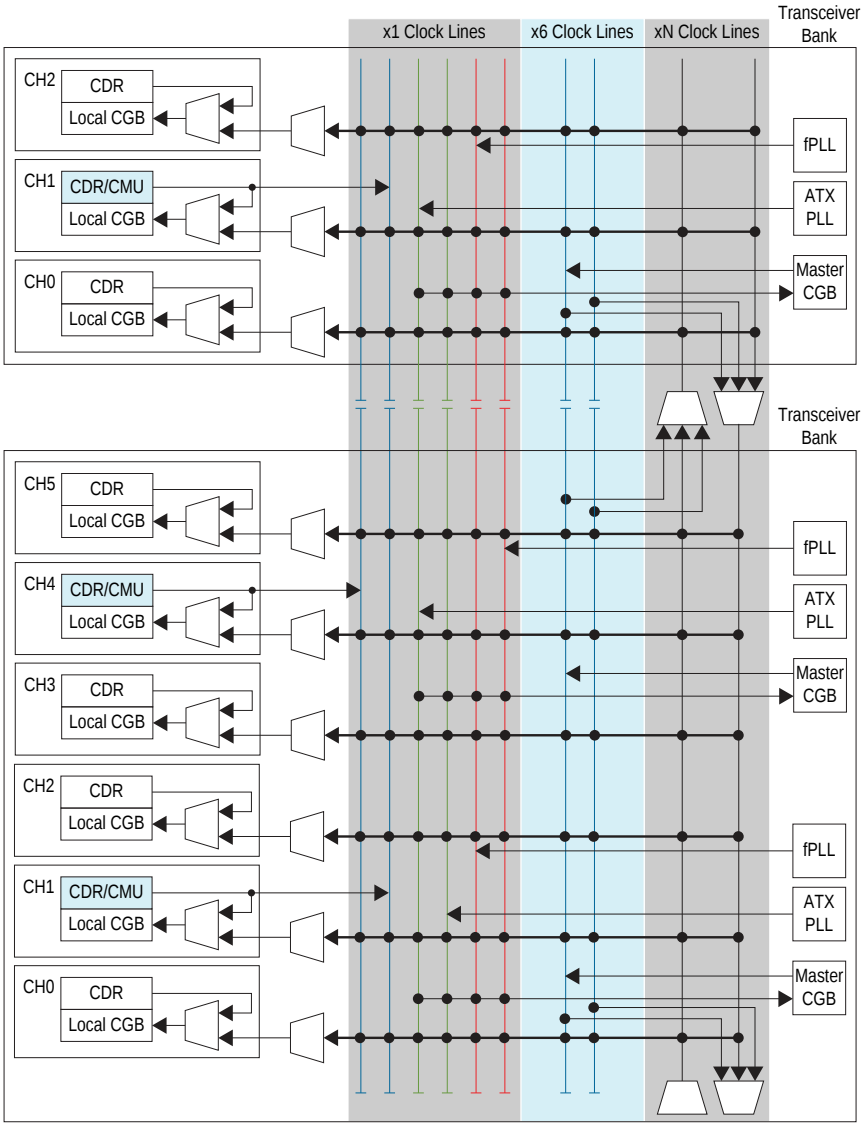
Transceiver banks can have either three or six transceiver channels. Transceiver banks with six transceiver channels have two advanced transmit (ATX) PLLs, fractional PLLs (fPLLs), and master clock generation blocks (CGBs). The transceiver banks with three transceiver channels have one master CGB, ATX PLL, and fPLL.

The Arria 10 transceiver clocking architecture supports both bonded and non-bonded transceiver channel configurations. Channel bonding is used to minimize the clock skew between multiple transceiver channels within the same interface. For Arria 10 transceivers, the term bonding can refer to PMA bonding as well as PMA and PCS bonding. Refer to the *Channel Bonding* section for more details.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 3-1: Arria 10 PLLs and Clock Networks



Related Information

Channel Bonding on page 3-38

PLLs

Table 3-1: Transmit PLLs in Arria 10 Devices

PLL Type	Data Rate Range[25]	Characteristics
Advanced Transmit (ATX) PLL	1.0 Gbps to 28.1 Gbps	• Best jitter performance • LC Tank based voltage controlled oscillator (VCO) • Supports fractional synthesis mode • Used for both bonded and non-bonded channel configurations
Fractional PLL (fPLL)	0.611 Gbps to 12.5 Gbps	• Ring oscillator based VCO • Supports fractional synthesis mode • Used for both bonded and non-bonded channel configurations
CMU (Clock Multiplier Unit) PLL / Channel PLL [26]	0.611 Gbps to 17.4 Gbps	• Ring oscillator based VCO • Used as an additional clock source for non-bonded applications

Note: For best overall performance, use the ATX PLL first, followed by the fPLL. The channel PLL can be used as a transmit PLL if additional PLLs are required.

Related Information

Clock Networks

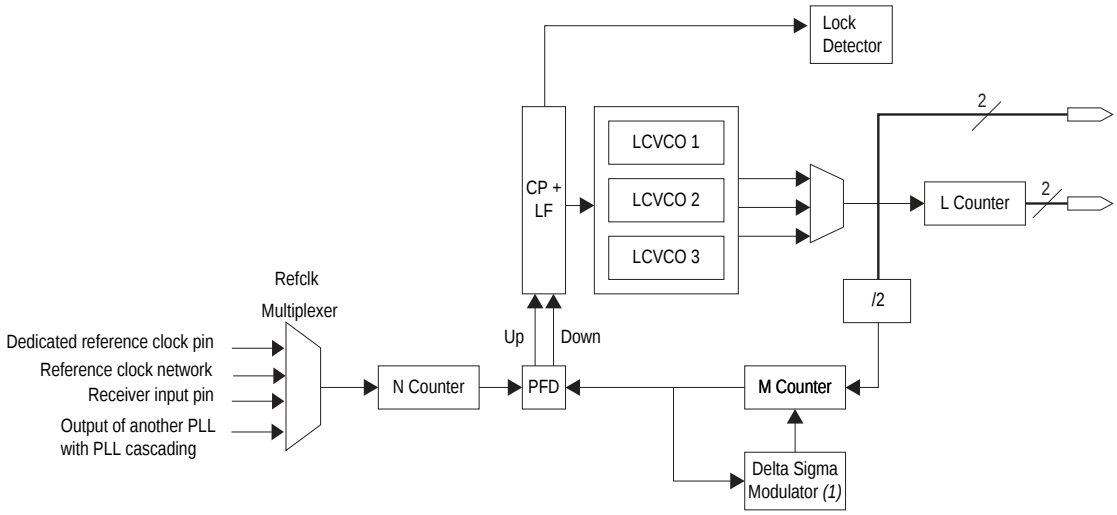
ATX PLL

The ATX PLL contains three LC - Tank based voltage controlled oscillators (VCOs). These three LC VCOs have different frequency ranges to support continuous range of operation. It supports both integral and fractional frequency synthesis.

The transceiver banks with six channels have two ATX PLLs each (one located at the top and the other at the bottom of the bank). The transceiver banks with three channels have only one ATX PLL.

[25] Data rates depend on characterization and may change after characterization report is available.
[26] The CMU PLL / Channel PLL of channel 1 and channel 4 only can be used as a transmit PLL.

Figure 3-2: ATX PLL Block Diagram



Note: (1) The Delta Sigma Modulator is enaged only when the ATX PLL is used in fractional mode.

Input Reference Clock

This is the dedicated input reference clock source for the PLL.

The input reference clock can be sourced from a dedicated reference clock pin, the reference clock network, a receiver input pin, or the output of another PLL with PLL cascading. The input reference clock is a differential signal. Altera recommends using the dedicated reference clock pin as the input reference clock source for the best jitter performance. The input reference clock must be stable and free-running at device power-up for proper PLL operation.

Reference Clock Multiplexer (Refclk Mux)

The refclk mux selects the reference clock to the PLL from the various reference clock sources available.

N Counter

The N counter divides the refclk mux's output. The division factors supported are 1, 2, 4, and 8.

Phase Frequency Detector (PFD)

The `refclk` signal at the output of the N counter block and the feedback clock `(fbclk)` signal at the output of the M counter block is supplied as an input to the PFD. The output of the PFD is proportional to the phase difference between the 2 inputs. It is used to align the `refclk` signal at the output of the N counter to the feedback clock `(fbclk)` signal. The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter (CP + LF)

The PFD output is used by the charge pump and loop filter to generate a control voltage for the VCO. The charge pump translates the "Up" / "Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage that drives the VCO frequency. The charge pump, loop filter, and VCO settings determine the bandwidth of the ATX PLL.

Lock Detector

The lock detector block indicates when the reference clock and the feedback clock are phase aligned. It generates an active high signal to indicate that the PLL is locked to its input reference clock.

Voltage Controller Oscillator (VCO)

The VCO used in ATX PLL is LC Tank based. The output of the charge pump and the loop filter serve as an input to the VCO. The output frequency of the VCO depends on the input control voltage. The output frequency is adjusted based on the output voltage of the charge pump and loop filter. Each ATX PLL has three LC tank circuits, and each tank circuit has multiple frequency banks in order to support a continuous frequency range of operation from 7 GHz up to 14.05 GHz.

L Counter

The L counter divides the differential clocks generated by the ATX PLL. The division factors supported are 1, 2, 4, 8, and 16. The L counter generates a differential clock, and is not in the feedback path of the PLL.

M Counter

The M counter's output is the same frequency as the N counter's output. The VCO frequency is governed by the equation:

VCO freq = M * refclk/N

Because the L-counter is not in the feedback path of the PLL, an additional divider is required to divide the high speed serial clock output of the VCO before it reaches the M counter. This divider divides by 2.

The division factors supported by the M counter are 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 30, 32, 36, 40, 48, 50, 60, 64, 80, and 100.

Delta Sigma Modulator

The delta sigma modulator is used only in fractional mode. It modulates the M counter divide value over time so that the PLL can perform fractional frequency synthesis.

Instantiating ATX PLL

The Arria 10 transceiver ATX PLL IP provides access to the ATX PLLs in hardware. One instance of the PLL IP represents one ATX PLL in hardware.

1. Open the Quartus II software
2. Click **Tools > MegaWizard Plug-In Manager**
3. Select the **Arria 10** device family
4. Select **PLL** in the **Installed Plug-Ins tree**. Choose **Arria 10 Transceiver ATX PLL**.
5. In **What name do you want for the output file?**, browse to the location where you want to save your design, and enter a filename. Click **Next**.

The ATX PLL IP MegaWizard Plug-In Manager window opens.

ATX PLL IP

The ATX PLL configuration options and parameters are listed in the following tables:

Table 3-2: ATX PLL Parameters and Settings

Parameter	Range	Description
Message level for rule violations	Error Warning	Specifies the messaging level to use for parameter rule violations. • Error—Causes all rule violations to prevent IP generation. • Warning—Displays all rule violations as warnings and will allow IP generation in spite of violations.
Device speed grade	fastest	Specifies the desired device speed grade. This information is used for PLL frequency validation.
Protocol mode	Basic PCIe Gen1 PCIe Gen2 PCIe Gen3	Governs the internal setting rules for the VCO. This parameter is not a preset. You must set all other parameters for your protocol.
Bandwidth	Low Medium High	Specifies the VCO bandwidth. Higher bandwidth reduces PLL lock time, at the expense of decreased jitter rejection.
Number of PLL reference clocks	1 to 5	Specifies the number of input reference clocks for the ATX PLL. Generally used for data rate reconfiguration.
Selected reference clock source	0 to 4	Specifies the initially selected reference clock input to the ATX PLL.
Primary PLL clock output buffer	GX clock output buffer GT clock output buffer	Specifies which PLL output is active initially. • If GX is selected, turn ON **"Enable PLL GX clock output port"** . • If GT is selected, turn ON **"Enable PLL GT clock output port"**
Enable PLL GX clock output port (27)	On/Off	Enables the GX output port which feeds x1 clock lines. Must be selected for PLL output frequency less than 8 GHz, or if reconfiguration below 8 GHz is planned. Turn ON this port if GX is selected in the **"Primary PLL clock output buffer"**,

(27) You can enable both the GX clock output port and the GT clock output port. However, only one port can be in operation at any given time. You can switch between the two ports using PLL reconfiguration.

Parameter	Range	Description
Enable PLL GT clock output port [27]	On/Off	Enables the GT output port which feeds dedicated high speed clock lines. Must be selected for PLL output frequency greater than 8 GHz, or if reconfiguration above 8 GHz is planned. Turn ON this port if GT is selected in the **"Primary PLL clock output buffer"** parameter.
Enable PCIe clock output port	On/Off	Exposes the `pll_pcie_clk` port used for PCI Express.
PLL output frequency		Specify the target output frequency for the PLL.
PLL reference clock frequency	100 MHz to 800 MHz	Selects the input reference clock frequency for the PLL.
Effective M Counter	Read only	Displays the effective M-counter value
Divide factor (N-Counter)	Read only	Displays the effective N-counter value
Divide factor (L-Counter)	Read only	Displays the effective L-counter value

Table 3-3: ATX PLL—Master Clock Generation Block Parameters and Settings

Parameter	Range	Description
Include Master Clock Generation Block [28]	On/Off	When enabled, includes a master CGB as a part of the ATX PLL IP. The PLL output drives the Master CGB. This is used for x6/xN bonded and non-bonded modes.
Clock division factor	1, 2, 4, 8	Divides the master CGB clock input before generating bonding clocks.
Enable x6/xN non-bonded high-speed clock output port	On/Off	Enables the master CGB serial clock output port used for x6/xN non-bonded modes
Enable PCIe clock switch interface	On/Off	Enables the control signals for the PCIe clock switch circuitry. Used for PCIe clock rate switching.
Number of auxiliary MCGB clock input ports	0, 1	Auxiliary input is used for PCIe Gen3.
MCGB input clock frequency	Read only	Displays the master CGB's required input clock frequency.
MCGB output data rate.	Read only	Displays the master CGB's output data rate.

[28] You have to manually enable the MCGB for bonding applications.

Parameter	Range	Description
Enable bonding clock output ports	On/Off	Enables the `tx_bonding_clocks` output ports of master CGB used for channel bonding. This option should be turned ON for bonded designs.
Enable feedback compensation bonding output port	On/Off	Enables the feedback output path of the master CGB used for feedback compensation bonding.
PMA interface width	8, 10, 16, 20, 32, 40, 64	Specifies PMA-PCS interface width. Proper value must be selected for generating bonding clocks for Native PHY IP. This value should match the PMA interface width selected for the Native PHY IP.

Table 3-4: ATX PLL—Dynamic Reconfiguration

Parameter	Range	Description
Enable reconfiguration	On/Off	Enables the PLL reconfiguration interface. Enables the simulation models and adds more ports for reconfiguration.
Configuration file prefix		Enter the prefix name for the configuration files to be generated.
Generate SystemVerilog package file	On/Off	Generates a SystemVerilog package file containing all relevant parameters used by the PLL.
Generate C header file	On/Off	Generates a C header file containing all relevant parameters used by the PLL.
Generate MIF (Memory Initialize File)	On/Off	Generates a MIF file which contains the current configuration. Use this option for reconfiguration purposes in order to switch between different PLL configurations.

Table 3-5: ATX PLL—Generation Options

Parameter	Range	Description
Generate parameter documentation file	On/Off	Generates a **.csv** file which contains descriptions of ATX PLL IP parameters and values.

Table 3-6: ATX PLL IP Ports

Parameter	Range	Clock Domain	Description
`pll_powerdown`	input	Asynchronous	Resets the PLL when asserted high.

Parameter	Range	Clock Domain	Description
pll_refclk0	input	N/A	Reference clock input port 0. There are a total of five reference clock input ports. The number of reference clock ports available depends on the **Number of PLL reference clocks** parameter.
pll_refclk1	input	N/A	Reference clock input port 1
pll_refclk2	input	N/A	Reference clock input port 2
pll_refclk3	input	N/A	Reference clock input port 3
pll_refclk4	input	N/A	Reference clock input port 4
tx_serial_clk	output	N/A	High speed serial clock output port for GX channels. Represents the x1 clock network.
tx_serial_clk_gt	output	N/A	High speed serial clock output port for GT channels. Represents the x1 clock network.
pll_locked	output	Asynchronous	Active high status signal which indicates if PLL is locked.
pll_pcie_clk	output	N/A	Used for PCIe.
reconfig_clk0	input	N/A	Optional Avalon interface clock. Used for PLL reconfiguration. The reconfiguration ports appear only if the **Enable Reconfiguration** parameter is selected in the PLL IP GUI. When this parameter is not selected, the ports are set to OFF internally.
reconfig_reset0	input	reconfig_clk0	Used to reset the Avalon interface.
reconfig_write0	input	reconfig_clk0	Active high write enable signal.
reconfig_read0	input	reconfig_clk0	Active high read enable signal.
reconfig_address0[9:0]	input	reconfig_clk0	10-bit address bus used to specify address to be accessed for both read and write operations.
reconfig_writedata0[31:0]	input	reconfig_clk0	32-bit data bus. Carries the write data to the specified address.

Parameter	Range	Clock Domain	Description
reconfig_readdata0[31:0]	output	reconfig_clk0	32-bit data bus. Carries the read data from the specified address.
reconfig_waitrequest0	output	reconfig_clk0	Indicates when the Avalon interface signal is busy. When asserted, all inputs must be held constant.
pll_cal_busy	output	Asynchronous	Status signal which is asserted high when PLL calibration is in progress. OR this signal with tx_cal busy port before connecting to the reset controller IP.
mcgb_rst	input	Asynchronous	Master CGB reset control. If pll feedback compensation bonding mode is used, deassert this reset at the same time as pll_powerdown. If pll feedback compensation bonding is not being used, then this port can be deasserted after pll_powerdown is de-asserted, but before tx_analogreset is deasserted. Alternatively, this port can be deasserted at the same time as pll_powerdown.
mcgb_aux_clk0	input	N/A	Used for PCIe implementation to switch between fPLL/ATX PLL during link speed negotiation.
tx_bonding_clocks[5:0]	Output	N/A	Optional 6-bit bus which carries the low speed parallel clock outputs from the master CGB. Used for channel bonding, and represents the x6/xN clock network
mcgb_serial_clk	Output	N/A	High speed serial clock output for x6/xN non-bonded configurations.
pcie_sw[1:0]	input	N/A	2-bit PCIe rate switch control input.
pcie_sw_done[1:0]	output	N/A	2-bit PCIe rate switch status output.

Related Information

Avalon Interface - Specifications

The ports related to reconfiguration are compliant with the Avalon Specification. Refer to the Avalon Specification for more details about these ports.

fPLL

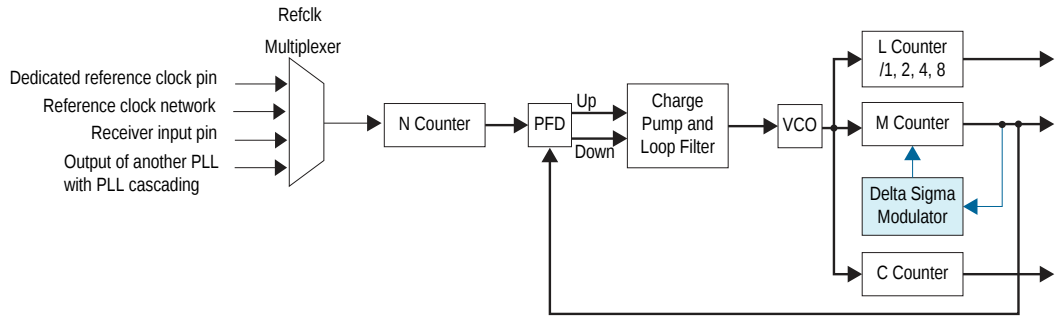
The fractional PLL (fPLL) is used for generating lower clock frequencies. It can support both integer and fractional frequency synthesis. The fractional PLL (fPLL) can be used as a transmit PLL for transceiver applications. It can be cascaded to the ATX or to another fPLL, or it can be used to drive the core clock network.

There are two fPLLs in each transceiver bank with six channels (one located at the top and the other at the bottom of the bank). Transceiver banks with three channels have only one fPLL.

For transceiver and PLL cascading applications, the fPLL can support continuous data rates from 611 Mbps to 12.5 Gbps in both integer and fractional frequency synthesis modes. PLL cascading enables additional flexibility in terms of reference clock selection.

When used to drive the FPGA fabric's core clock network, the fPLL can support frequencies from 27 MHz up to the core clock network's maximum frequency (f_{MAX}).

Figure 3-3: fPLL Block Diagram



The fPLL generates output clocks with a fixed frequency and phase relation to an input reference clock. The resolution of the fractional loop counter in the fPLL is ½ and it can support fractional frequency modes for data rates from 611 Mbps to 12.5 Gbps.

Input Reference Clock

This is the dedicated input reference clock source for the PLL.

The input reference clock can be sourced from either a dedicated reference clock pin, or the reference clock network, or a receiver input pin, or the output of another PLL with PLL cascading. The input reference clock is a differential signal. Altera recommends using the dedicated reference clock pin as the input reference clock source for the best jitter performance. The input reference clock must be stable and free-running at device power-up for proper PLL operation.

Reference Clock Multiplexer (Refclk Mux)

The refclk mux selects the reference clock to the PLL from the various reference clock sources available.

N Counter

The N counter divides the reference clock (refclk) mux's output. The N counter division helps lower the loop bandwidth or reduce the frequency within the PFD operating range. The N counter supports division factors from 1 to 32.

Phase Frequency Detector (PFD)

The reference clock (`refclk`) signal at the output of the N counter block and the feedback clock (`fbclk`) signal at the output of the M counter block is supplied as an input to the PFD. The output of the PFD is proportional to the phase difference between the two inputs. It is used to align the reference clock(`refclk`) to the feedback clock (`fbclk`). The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when the feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter (CP + LF)

The PFD output is used by the charge pump and loop filter to generate a control voltage for the VCO. The charge pump translates the "Up" / "Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage that drives the VCO frequency.

Voltage Controlled Oscillator (VCO)

The fPLL has a ring oscillator based VCO. The VCO transforms the input control voltage into an adjustable frequency clock.

VCO freq = M * refclk/N. (N and M are the N counter and M counter division factors.)

L Counter

The L counter divides the VCO's clock output. When fPLL acts as a transmit PLL, the output of the L counter drives the clock generation block (CGB) and the TX PMA. The division factors supported are 1, 2, 4, and 8.

M Counter

The M counter divides the VCO's clock output. The M counter can select any VCO phase. The output of the M counter and N counter have same frequency.

Delta Sigma Modulator

The delta sigma modulator is used in fractional mode. Depending on the value of K input, it modulates the output of M counter over time. The value of K input can be changed dynamically to use the fPLL as replacement for the VCXO.

The delta sigma modulator can be configured in 1st order, 2nd order, or 3rd order mode.

C Counters

The C counter's design is identical to the M counter's design. However, the C counter is present in the fPLL's output path and is not in the PLL's feedback path.

Dynamic Phase Shift

The dynamic phase shift block allows the user to adjust the phase of the M and C counters in user mode. In fractional mode, dynamic phase shift is only available for the C counters.

Latency

The fPLL contains a 1 ns delay with 50 ps resolution on each C, M, and N counter. In addition, there is a 7 ns delay with 1 ns resolution on both the reference clock and feedback clock paths. The C and M counters can be configured to select any VCO phase and a delay of up to 128 clock cycles. The selected VCO phase can be changed dynamically. The M counter's phase cannot be changed in fractional mode.

Instantiating fPLL IP

The fPLL IP for Arria 10 transceivers provides access to fPLLs in hardware. One instance of fPLL IP represents one fPLL in hardware.

1. Open the Quartus II software
2. Click **Tools > MegaWizard Plug-In Manager**
3. Select the **Arria 10** device family
4. Select **PLL** in the **Installed Plug-Ins tree**. Choose **Arria 10 fPLL**.
5. In **What name do you want for the output file?**, browse to the location where you want to save your design, and enter a filename. Click **Next**.

The fPLL IP MegaWizard Plug-In Manager window opens.

fPLL IP

The fPLL configuration options and parameters are listed in the following tables.

Table 3-7: fPLL Parameters and Settings

Parameters	Range	Description
Device speed grade	fastest	Specifies the desired device speed grade. This information is used for PLL frequency validation.
Reference clock frequency	27 MHz to 700 MHz	Specifies the PLL input reference clock frequency.
Bandwidth	Auto Low Medium High	Specifies the VCO bandwidth. Higher bandwidth reduces PLL lock time, at the expense of decreased jitter rejection.
Number of PLL reference clocks	1 to 5	Specifies the number of input reference clocks for the fPLL. Generally used for data rate reconfiguration.
Selected reference clock source	0 to 4	Specifies the initially selected reference clock input to the fPLL.
Enable fractional mode	On/Off	Enables the fractional frequency synthesis mode. This enables the PLL to output frequencies which are not integral multiples of the input reference clock.
Enable physical output clock parameters	On/Off	
Operation mode	Direct Normal IQTXRXCLK	Specifies the feedback operation mode for the fPLL.
Enable PCIe clock output port	On/Off	Exposes the `pll_pcie_clk` port used for PCIe.

Parameters	Range	Description
Enable cascade clock output port (fPLL to fPLL cascading)	On/Off	Enables the cascade source output port for fPLL to fPLL cascading. Connect this port to the `refclk` input port of the cascaded PLL.
Specifies which core outclk to be used as cascading source	0-3	Specifies the cascading source for fPLL to fPLL cascading.
Enable cascade clock output port (fPLL to ATX PLL cascading)	On/Off	Enables the cascade source output port for fPLL to ATX PLL cascading. Connect this port to the `refclk` input port of the cascaded PLL.
Specifies which core outclk to be used as cascading source	0-3	Specifies the cascading source for fPLL to ATX PLL cascading.
Enable access to dynamic phase shift ports	On/Off	Enables access to dynamic phase shift ports. When this option is selected, `phase_reset`, `phase_en`, `updn`, `cntsel[3:0]`, `num_phase_shifts[2:0]`, `reconfig_avmm`, `phase_done` ports are enabled.
Use as Transceiver PLL	On/Off	Use fPLL to drive transceiver channels through the transceiver clock network
Protocol Mode	Basic PCIe Gen1 PCIe Gen2	Governs the internal setting rules for the VCO. This parameter is not a preset. You must set all other parameters for your protocol.
PLL output frequency		Specifies the target output frequency for the PLL
Use as Core PLL	On/Off	Use fPLL as a general purpose PLL to drive the FPGA core clock network.
Number of Clocks	1 to 4	Specifies the number of output clocks.
Desired Frequency		Enter the desired output clock frequency.
Actual Frequency		Specifies the output clock frequency.
Phase shift units		Specifies the unit used for phase shift measurement. Select the appropriate unit.
Phase shift		Specifies the requested phase shift value. Enter the desired phase shift value.
Actual phase shift		Specifies the actual phase shift value.

Table 3-8: fPLL—Master Clock Generation Block Parameters and Settings

Parameters	Range	Description
Include Master Clock Generation Block	On/Off	When enabled, includes a master CGB as a part of the fPLL IP. The PLL output drives the master CGB. This is used for x6/xN bonded and non-bonded modes.
Clock division factor	1, 2, 4, 8	Divides the master CGB clock input before generating bonding clocks.
Enable x6/xN non-bonded high-speed clock output port	On/Off	Enables the master CGB serial clock output port used for x6/xN non-bonded modes.
Enable PCIe clock switch interface	On/Off	Enables the control signals used for PCIe clock switch circuitry.
Number of auxiliary MCGB clock input ports		Auxiliary input is intended for PCIe Gen3
MCGB input clock frequency	Read only	Displays the master CGB's required input clock frequency. You cannot set this parameter.
MCGB output data rate	Read only	Displays the master CGB's output data rate. You cannot set this parameter. This value is calculated based on MCGB input clock frequency and MCGB clock division factor.
Enable bonding clock output ports	On/Off	Enables the `'tx_bonding_clocks'` output ports of the Master CGB used for channel bonding. You must enable this for bonded designs.
Enable feedback compensation bonding	On/Off	Enables the feedback output path of the master CGB used for feedback compensation bonding. When enabled, the feedback connections are automatically handled by the PLL IP.
PMA interface width	8, 10, 16, 20, 32, 40, 64	Specifies PMA-PCS interface width. Proper value must be selected for generating bonding clocks for Native PHY IP. This value should match the PMA interface width selected for the Native PHY IP.

Table 3-9: fPLL—Dynamic Reconfiguration Parameters and Settings

Parameter	Range	Description
Enable reconfiguration	On/Off	Enables the PLL reconfiguration interface.
Configuration file prefix		Enter the prefix name for the configuration files to be generated.

Parameter	Range	Description
Generate SystemVerilog package file	On/Off	Generates a SystemVerilog package file containing all relevant parameters used by the PLL.
Generate C header file	On/Off	Generates a C header file containing all relevant parameters used by the PLL.
Generate MIF (Memory Initialize File)	On/Off	Generates a MIF file that contains the current configuration. Use this option for reconfiguration purposes in order to switch between different PLL configurations.

Table 3-10: fPLL—Clock Switch over Parameters and Settings

Parameter	Range	Description
Create a second input clock `'pll_refclk1'`	On/Off	Turn on this parameter to have a secondary clock attached to your fPLL that can switch with your original reference clock.
Second Reference Clock Frequency		Specifies the 2nd reference clock frequency for the fPLL.
Switch over Mode	Automatic Manual Automatic with Manual Override	Specifies how input frequency switch over will be handled. • Automatic—will use built in circuitry to detect if one of your input clocks has stopped toggling and switch to another one. • Manual—will create an `EXTSWITCH` signal which can be used to manually switch the clock by asserting high for at least 3 clock cycles. • Automatic with Manual Override—behaves as an **Automatic Switch over** until the `EXTSWITCH` goes high, in which case it will switch and ignore any automatic switches as long as `EXTSWITCH` stays high.
Switch over Delays	0 to 7	Delays the switch over process by the specified number of cycles.
Create an `'active_clk'` signal to indicate the input clock in use	On/Off	Creates an output which indicates the input clock currently in use by the PLL. A logic Low on this signal indicates `refclk0` is in use and a logic High indicates `refclk1` is in use.
Create a `'clkbad'` signal for each of the input clocks	On/Off	Generates 2 CLKBAD outputs, one for each input clock. A logic Low indicates the `CLK` is present and a logic High indicates the `CLK` is absent.

Table 3-11: fPLL - Generation Options

Parameter	Range	Description
Generates parameter documentation file	On/Off	Generates a `.csv` file that contains descriptions of all the fPLL parameters and values.

Table 3-12: fPLL IP Ports

Parameter	Range	Clock Domain	Description
`pll_powerdown`	input	Asynchronous	Resets the PLL when asserted high.
`pll_refclk0`	input	N/A	Reference clock input port 0. There are five reference clock input ports. The number of reference clock ports available depends on the **Number of PLL reference clocks** parameter.
`pll_refclk1`	input	N/A	Reference clock input port 1
`pll_refclk2`	input	N/A	Reference clock input port 2
`pll_refclk3`	input	N/A	Reference clock input port 3
`pll_refclk4`	input	N/A	Reference clock input port 4
`tx_serial_clk`	output	N/A	High speed serial clock output port for GX channels. Represents the x1 clock network.
`tx_serial_clk_n`	output	N/A	This is `tx_serial_clk` with 180 degree phase shift.
`pll_locked`	output	Asynchronous	Active high status signal which indicates if PLL is locked.
`pll_pcie_clk`	output	N/A	Used for PCIe.
`reconfig_clk0`	input	N/A	Optional Avalon interface clock. Used for PLL reconfiguration.
`reconfig_reset0`	input	`reconfig_clk0`	Used to reset the Avalon interface.
`reconfig_write0`	input	`reconfig_clk0`	Active high write enable signal.
`reconfig_read0`	input	`reconfig_clk0`	Active high read enable signal.
`reconfig_address0[9:0]`	input	`reconfig_clk0`	10-bit address bus used to specify address to be accessed for both read and write operations.

Parameter	Range	Clock Domain	Description
reconfig_writedata0[31:0]	input	reconfig_clk0	32-bit data bus. Carries the write data to the specified address.
reconfig_readdata0[31:0]	output	reconfig_clk0	32-bit data bus. Carries the read data from the specified address.
reconfig_waitrequest0	output	reconfig_clk0	Indicates when the Avalon interface signal is busy. When asserted, all inputs must be held constant.
pll_cal_busy	output	Asynchronous	Status signal which is asserted high when PLL calibration is in progress. OR this signal with the tx_cal_busy port on the reset controller IP.
mcgb_rst	input	N/A	Master CGB reset control. If pll feedback compensation bonding mode is used, deassert this reset at the same time as pll_powerdown. If pll feedback compensation bonding is not being used, then this port can be deasserted after pll_powerdown is deasserted, but before tx_analogreset is de-asserted. Alternatively, this port can be deasserted at the same time as pll_powerdown
mcgb_aux_clk0	input	N/A	Used for PCIe to switch between fPLL/ATX PLL during link speed negotiation
tx_bonding_clocks[5:0]	Output	N/A	Optional 6-bit bus which carries the low speed parallel clock outputs from the Master CGB. Used for channel bonding, and represents the x6/xN clock network
mcgb_serial_clk	Output	N/A	High speed serial clock output for x6/xN non-bonded configurations.
pcie_sw[1:0]	input	N/A	2-bit PCIe rate switch control input.

Parameter	Range	Clock Domain	Description
pcie_sw_done[1:0]	output	N/A	2-bit PCIe rate switch status output.
pll_cascade_clk	output	N/A	fPLL to fPLL cascading clock output port.
atx_pll_cascade_clk	output	N/A	fPLL to ATX PLL cascading clock output port.
activeclk	output		Creates an output signal that indicates the input clock being used by the PLL. A logic Low on this signal indicates refclk0 is being used and a logic High indicates refclk1 is being used.

Related Information

Avalon Interface - Specifications

The ports related to reconfiguration are compliant with the Avalon Specification. Refer to the Avalon Specification for more details about these ports.

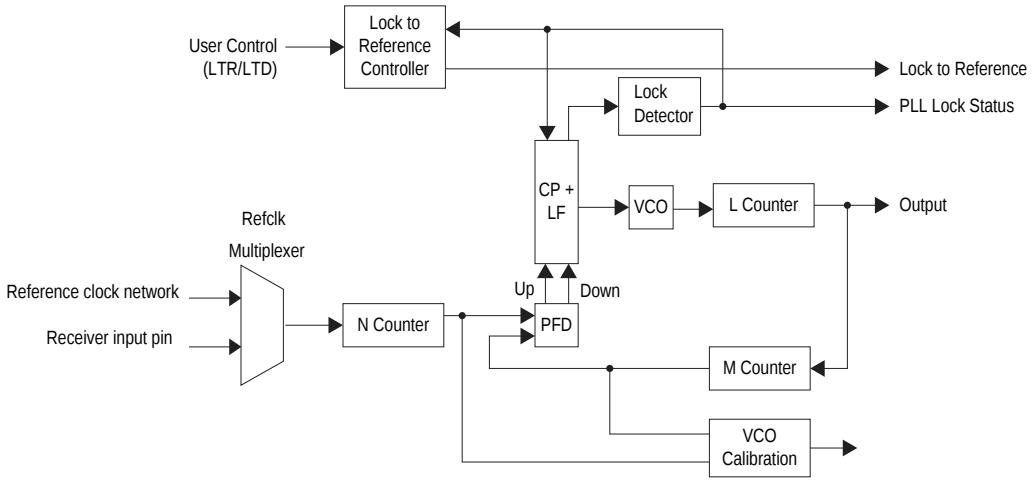
CMU PLL

The clock multiplier unit (CMU) PLL / channel PLL / CDR resides locally within each transceiver channel. The channel PLL's primary function is to recover the receiver clock and data in the transceiver channel. In this case the PLL is used in clock and data recovery (CDR) mode.

When the channel PLL of channel 1 and 4 is configured in the CMU mode, the channel PLL can drive the local clock generation block (CGB) of its own channel. However, when the channel PLL is used as a CMU PLL, the channel can only be used as a transmitter channel as the CDR block is not available to recover the received clock and data.

The CMU PLL from transceiver channel 1 and channel 4 can be used to drive other transceiver channels within the same transceiver bank. The CDR of channels 0, 2, 3 and 5 cannot be configured as a CMU PLL.

Figure 3-4: CMU PLL Block Diagram



Input Reference Clock

The input reference clock for a CMU PLL can be sourced from either the reference clock network or a receiver input pin. The input reference clock is a differential signal. The input reference clock must be stable and free-running at device power-up for proper PLL operation.

Reference Clock Multiplexer (Refclk Mux)

The refclk mux selects the input reference clock to the PLL from the various reference clock sources available.

N Counter

The N counter divides the refclk mux's output. The N counter division helps lower the loop bandwidth or reduce the frequency to within the phase frequency detector's (PFD) operating range. Possible divide ratios are 1 (bypass), 2, 4, and 8.

Phase Frequency Detector (PFD)

The reference clock (`refclk`) signal at the output of the N counter block and the feedback clock (`fbclk`) signal at the output of the M counter block is supplied as an input to the PFD. The PFD output is proportional to the phase difference between the two inputs. It aligns the input reference clock(`refclk`) to the feedback clock (`fbclk`). The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter (CP + LF)

The PFD output is used by the charge pump and loop filter to generate a control voltage for the VCO. The charge pump translates the "Up" / "Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage which drives the VCO frequency.

Voltage Controlled Oscillator (VCO)

The CMU PLL has a ring oscillator based VCO. The fundamental VCO frequency range is from 4 GHz to 14 GHz. Lower frequencies can be generated using the PFD and M counter settings.

L Counter

The L counter divides the differential clocks generated by the CMU PLL. The division factors supported are 1, 2, 4, and 8.

M Counter

The M counter is used in the PFD's feedback path. The output of the L counter is connected to the M counter. The combined division ratios of the L counter and the M counter determine the overall division factor in the PFD's feedback path.

The division factors supported are 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 30, 32, 36, 40, 48, 50, 60, 64, 72, 80, 96, 100, 120, 128, 160, and 200.

Lock Detector (LD)

The lock detector indicates when the CMU PLL is locked to the desired output's phase and frequency. The lock detector XORs the "Up" / "Down" pulses and indicates when the M counter's output and N counter's output are phase-aligned.

The reference clock (`refclk`) and feedback clock (`fbclk`) are sent to the PCS' ppm detector block. There is a pre-divider to lower the frequency in case the frequency is too high.

Instantiating CMU PLL IP

The CMU PLL IP for Arria 10 transceivers provides access to the CMU PLLs in hardware. One instance of the PLL IP represents one CMU PLL in hardware.

1. Open the Quartus II software
2. Click **Tools > MegaWizard Plug-In Manage**r
3. Select the **Arria 10** device family
4. Select **PLL** in the **Installed Plug-Ins tree**. Choose **Arria 10 Transceiver CMU PLL**.
5. In **What name do you want for the output file?**, browse to the location where you want to save your design, and enter a filename. Click **Next**.

The CMU PLL IP MegaWizard Plug-In Manager window opens.

CMU PLL IP

The CMU PLL configuration options and parameters are listed in the following tables.

Table 3-13: CMU PLL Parameters and Settings

Parameters	Range	Description
Device speed grade	fastest	Specifies the desired device speed grade. This information is used for PLL frequency validation.
Protocol mode	Basic	Governs the internal setting rules for the VCO. This parameter is not a preset. You must set all other parameters for your protocol.
Bandwidth	Low Medium High	Specifies the VCO bandwidth. Higher bandwidth reduces PLL lock time, at the expense of decreased jitter rejection.
Number of PLL reference clocks	1 to 5	Specifies the number of input reference clocks for the CMU PLL. Generally used for data rate reconfiguration.

Parameters	Range	Description
Selected reference clock source	0 to 4	Specifies the initially selected reference clock input to the CMU PLL.
Message level for rule violations	Error Warning	Specifies the messaging level to use for parameter rule violations. • Error—Causes all rule violations to prevent IP generation. • Warning—Displays all rule violations as warnings and will allow IP generation in spite of violations.
PLL output frequency		Specify the target output frequency for the PLL.
PLL reference clock frequency	50 MHz to 800 MHz	Selects the input reference clock frequency for the PLL.
Multiply factor (M-Counter)	Read only	Displays the effective M-divider value
Divide factor (N-Counter)	Read only	Displays the effective N-counter value.
Divide factor (L-Counter)	Read only	Displays the effective L-counter value.

Table 3-14: CMU PLL—Dynamic Reconfiguration

Parameters	Range	Description
Enable reconfiguration	On/Off	Enables the PLL reconfiguration interface. Enables the simulation models and adds more ports for reconfiguration.
Configuration file prefix		Enter the prefix name for the configuration files to be generated.
Generate SystemVerilog package file	On/Off	Generates a SystemVerilog package file containing all relevant parameters used by the PLL.
Generate C header file	On/Off	Generates a C header file containing all relevant parameters used by the PLL.
Generate MIF (Memory Initialize File)	On/Off	Generates a MIF file that contains the current configuration. Use this option for reconfiguration purposes in order to switch between different PLL configurations.

Table 3-15: CMU PLL—Generation Options

Parameters	Range	Description
Generate parameter documentation file	On/Off	Generates a **.csv** file which contains the descriptions of all CMU PLL parameters and values.

Table 3-16: CMU PLL IP Ports

Parameter	Range	Clock Domain	Description
pll_powerdown	input	Asynchronous	Resets the PLL when asserted high.
pll_refclk0	input	N/A	Reference clock input port 0. There are 5 reference clock input ports. The number of reference clock ports available depends on the **Number of PLL reference clocks** parameter.
pll_refclk1	input	N/A	Reference clock input port 1
pll_refclk2	input	N/A	Reference clock input port 2
pll_refclk3	input	N/A	Reference clock input port 3
pll_refclk4	input	N/A	Reference clock input port 4
tx_serial_clk	output	N/A	High speed serial clock output port for GX channels. Represents the x1 clock network.
pll_locked	output	Asynchronous	Active high status signal which indicates if PLL is locked.
reconfig_clk0	input	N/A	Optional Avalon interface clock. Used for PLL reconfiguration. The reconfiguration ports appear only if the **Enable Reconfiguration** parameter is selected in the PLL IP GUI. When this parameter is not selected, the ports are set to OFF internally.
reconfig_reset0	input	reconfig_clk0	Used to reset the Avalon interface.
reconfig_write0	input	reconfig_clk0	Active high write enable signal.
reconfig_read0	input	reconfig_clk0	Active high read enable signal.
reconfig_address0[9:0]	input	reconfig_clk0	10-bit address bus used to specify address to be accessed for both read and write operations.

Parameter	Range	Clock Domain	Description
reconfig_writedata0[31:0]	input	reconfig_clk0	32-bit data bus. Carries the write data to the specified address.
reconfig_readdata0[31:0]	output	reconfig_clk0	32-bit data bus. Carries the read data from the specified address.
reconfig_waitrequest0	output	reconfig_clk0	Indicates when the Avalon interface signal is busy. When asserted, all inputs must be held constant.
pll_cal_busy	output	Asynchronous	Status signal that is asserted high when PLL calibration is in progress. Connect this port by OR-ing it with tx_cal_busy port on the reset controller IP

Related Information

Avalon Interface - Specifications

The ports related to reconfiguration are compliant with the Avalon Specification. Refer to the Avalon Specification for more details about these ports.

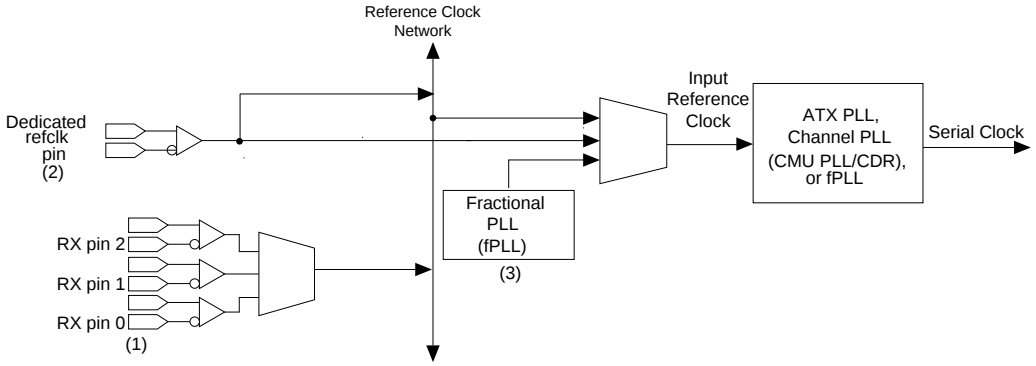
Input Reference Clock Sources

The transmitter PLL and the clock data recovery (CDR) block need an input reference clock source to generate the clocks required for transceiver operation. The input reference clock must be stable and free-running at device power-up for proper PLL operation.

Arria 10 transceiver PLLs have four possible input reference clock sources:

- Dedicated reference clock pins
- Receiver input pins
- The output of another PLL with PLL cascading
- Reference clock network

Figure 3-5: Input Reference Clock Sources



Note: (1) You can choose only one of the three RX pins to be used as an input reference clock source. Any RX pin on the same side of the device can be used as an input reference clock.

(2) Dedicated refclk pin can be used as an input reference clock source only for ATX or fPLL or to the reference clock network. Reference clock network can then drive the CMU PLL.

(3) The output of another PLL can be used as an input reference clock source during PLL cascading.

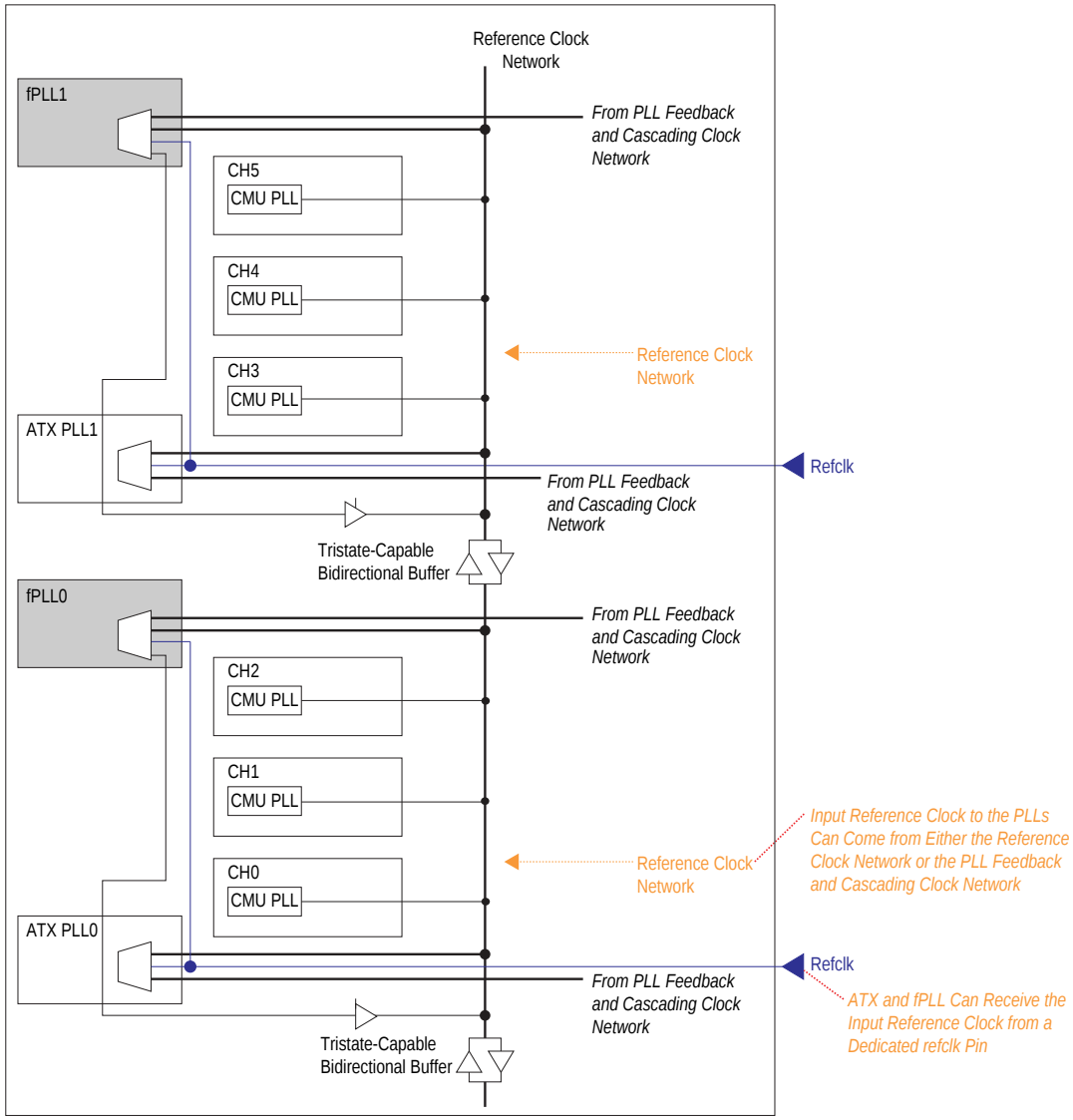
Note: In Arria 10 devices, the FPGA fabric core clock network can be used as an input reference source for fPLLs.

Dedicated Reference Clock Pins

To minimize the jitter, the advanced transmit (ATX) PLL and the fractional PLL (fPLL) can source the input reference clock directly from the reference clock buffer without passing through the reference clock network. The input reference clock is also fed into the reference clock network.

Figure 3-6: Dedicated Reference Clock Pins

There are two dedicated reference clock (refclk) pins available in each transceiver bank. The bottom refclk pin feeds the bottom ATX PLL and fPLL, and the top refclk pin feeds the top ATX PLL and fPLL.



Receiver Input Pins

Receiver input pins can be used as an input reference clock source.

The receiver input pin drives the reference clock network, which can then feed any number of transmitter PLLs on the same side of the device. When a receiver input pin is used as an input reference clock source, the clock data recovery (CDR) block of that channel is not available. As indicated in **Figure 3-5**, only one RX differential pin pair per three channels can be used as an input reference clock source at any given time.

PLL Cascading as a Input Reference Clock Source

In PLL cascading, PLL outputs are connected to the feedback and cascading clock network. The input reference clock to the first PLL can be sourced from the same network. In this mode, the output of one PLL

drives the reference clock input of another PLL. PLL cascading can generate frequency outputs not normally possible with a single PLL solution. The transceivers in Arria 10 devices support fPLL to fPLL or fPLL to ATX PLL cascading.

Reference Clock Network

The reference clock network distributes a reference clock source to either the entire left or right side of the FPGA where the transceivers reside. This allows any reference clock pin to drive any transmitter PLL on the same side of the device. Designs using multiple transmitter PLLs which require the same reference clock frequency and are located along the same side of the device, can share the same dedicated reference clock (refclk) pin.

Transmitter Clock Network

The transmitter clock network routes the clock from the transmitter PLL to the transmitter channel. It provides following two types of clocks to the transmitter channel:

- High Speed Serial clock – high speed clock for the serializer.
- Low Speed Parallel clock – low speed clock for the serializer and the PCS.

In a bonded channel configuration, both the serial clock and the parallel clock are routed from the transmitter PLL to the transmitter channel. In a non-bonded channel configuration, only the serial clock is routed to the transmitter channel, and the parallel clock is generated locally within the channel. To support various bonded and non-bonded clocking configurations, 4 types of transmitter clock network lines are available:

- x1 Clock Lines
- x6 Clock Lines
- xN Clock Lines
- GT Clock Lines

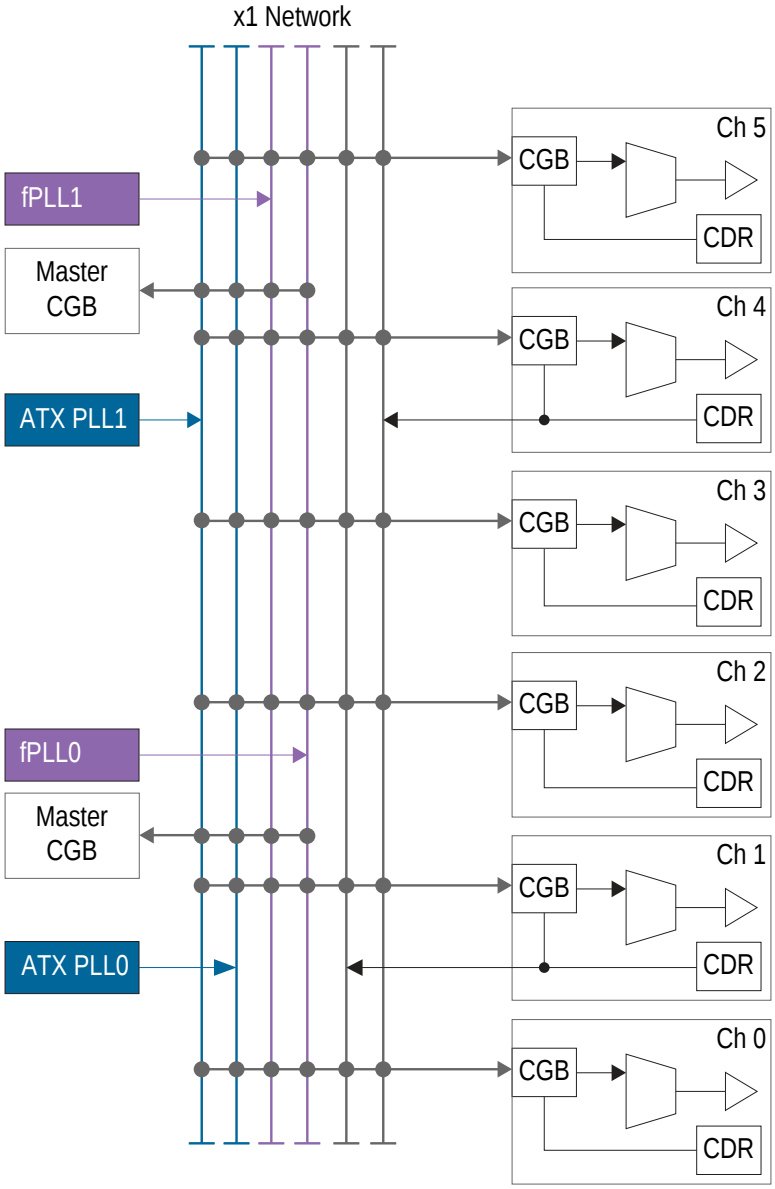
x1 Clock Lines

The x1 clock lines route the high speed serial clock output of a PLL to any channel within a transceiver bank. The low speed parallel clock is then generated by that particular channel's local clock generation block (CGB). Non-bonded channel configurations use the x1 clock network.

The x1 clock lines can be driven by the ATX PLL, fPLL, or by either one of the two channel PLLs (Channel 1 and 4 when used as a CMU PLL) within a transceiver bank. The channel PLL of channel 1 and channel 4 can be configured in CMU mode in order to act as a transmit PLL. When the channel PLL (of channel 1 and channel 4) is configured in the CMU mode, it can drive the local CGB of its own channel. However, because the CDR block is used as a transmit PLL, it is not available for clock and data recovery functions and the channel can only be used as a transmitter channel.

The x1 clock lines are also used to drive the master CGB in bonded channel configurations. Either one of the master CGB in each transceiver bank can drive the x6 clock lines for bonded channel configurations. The master CGB can only be driven by the ATX PLL or the fPLL. The CMU PLLs cannot drive the master CGB. Therefore, the CMU PLLs cannot be used for bonding purposes.

Figure 3-7: x1 Clock Lines



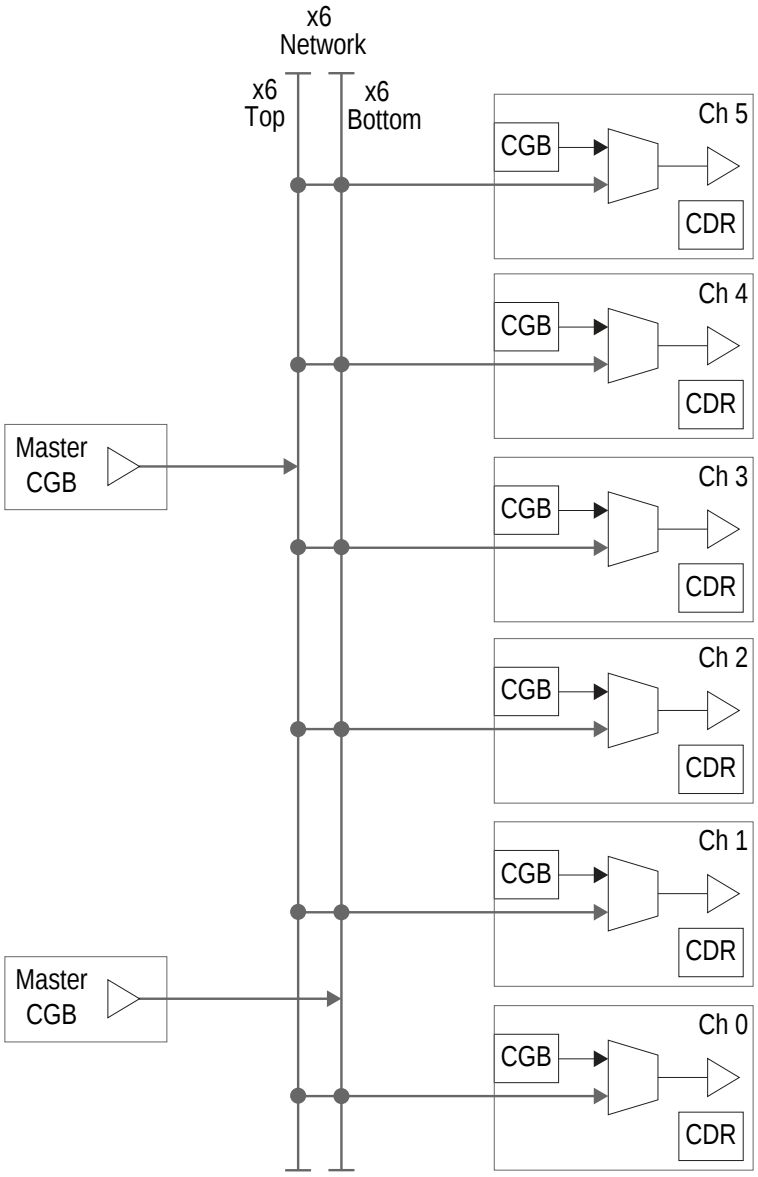
x6 Clock Lines

The x6 clock lines route the clock within a transceiver bank. The x6 clock lines are driven by the master CGB. There are two x6 clock lines per transceiver bank, one for each master CGB. Any channel within a transceiver bank can be driven by the x6 clock lines.

For bonded configuration mode, the low speed parallel clock output of the master CGB is used and the local CGB within each channel is bypassed. For non-bonded configurations, the master CGB can also provide a high speed serial clock output to each channel. In this case, the local CGB within each channel is not bypassed.

The x6 clock lines also drive the xN clock lines which route the clocks to the neighboring transceiver banks.

Figure 3-8: x6 Clock Lines

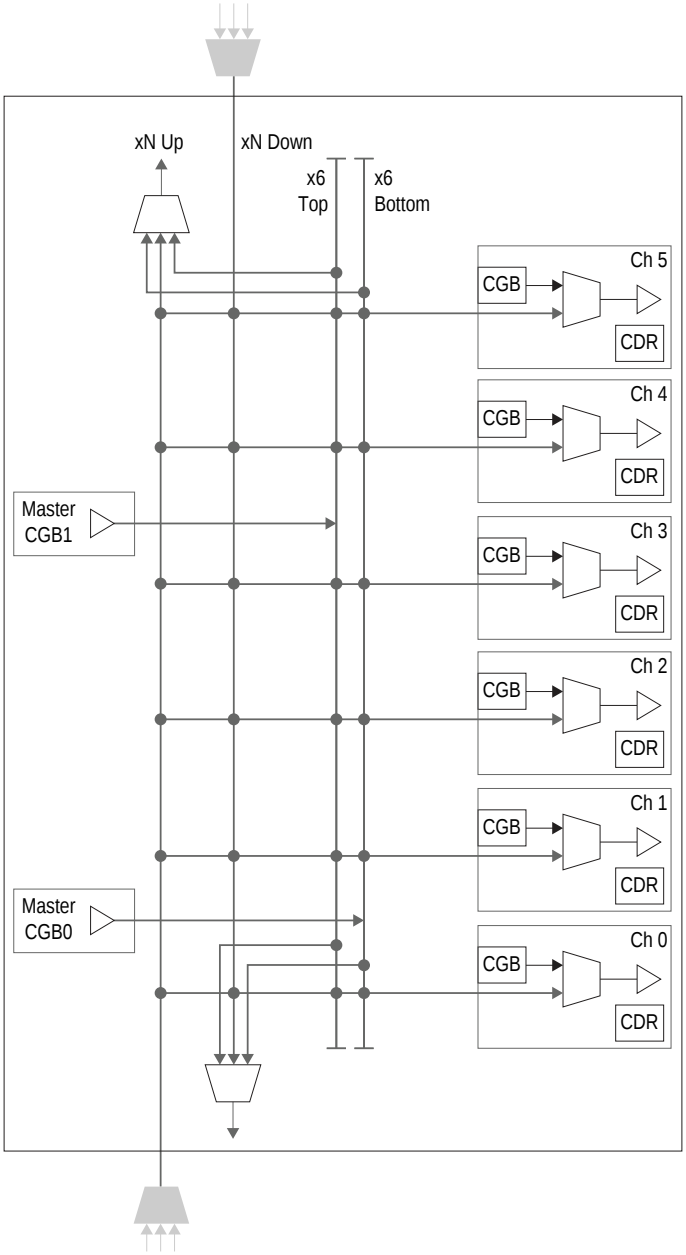


xN Clock Lines

The xN clock lines route the transceiver clocks across multiple transceiver banks.

The master CGB drives the x6 clock lines and the x6 clock lines drive the xN clock lines. There are two xN clock lines: xN Up and xN Down. xN Up clock lines route the clocks to transceiver banks located above the master CGB and xN Down clock lines route the clocks to transceiver banks located below the master CGB. The xN clock lines can be used in both bonded and non-bonded configurations. For bonded configurations, the low speed parallel clock output of the master CGB is used, and the local CGB within each channel is bypassed. For non-bonded configurations, the master CGB provides a high speed serial clock output to each channel.

Figure 3-9: xN Clock Network



The maximum channel span of xN clock network is two transceiver banks above and two transceiver banks below the bank that contains the driving PLL and the master CGB. Thus, a maximum of 30 channels can be used in a single bonded or non-bonded xN group.

The maximum data rate supported by the xN clock network while driving channels in either the bonded or non-bonded mode depends on the voltage used to drive the transceiver banks. All transceiver banks in a bonded group must share the same voltage. Refer to the table below for the maximum data rate supported.

Table 3-17: xN Clock Network Data Rate Restrictions

Transceiver Voltage	Maximum Data Rate Supported by xN Clock Network [29]
0.9 V	10.3125 Gbps
1.0 V	15 Gbps
1.1 V	16 Gbps

Related Information

x6/xN Bonding Mode on page 3-45

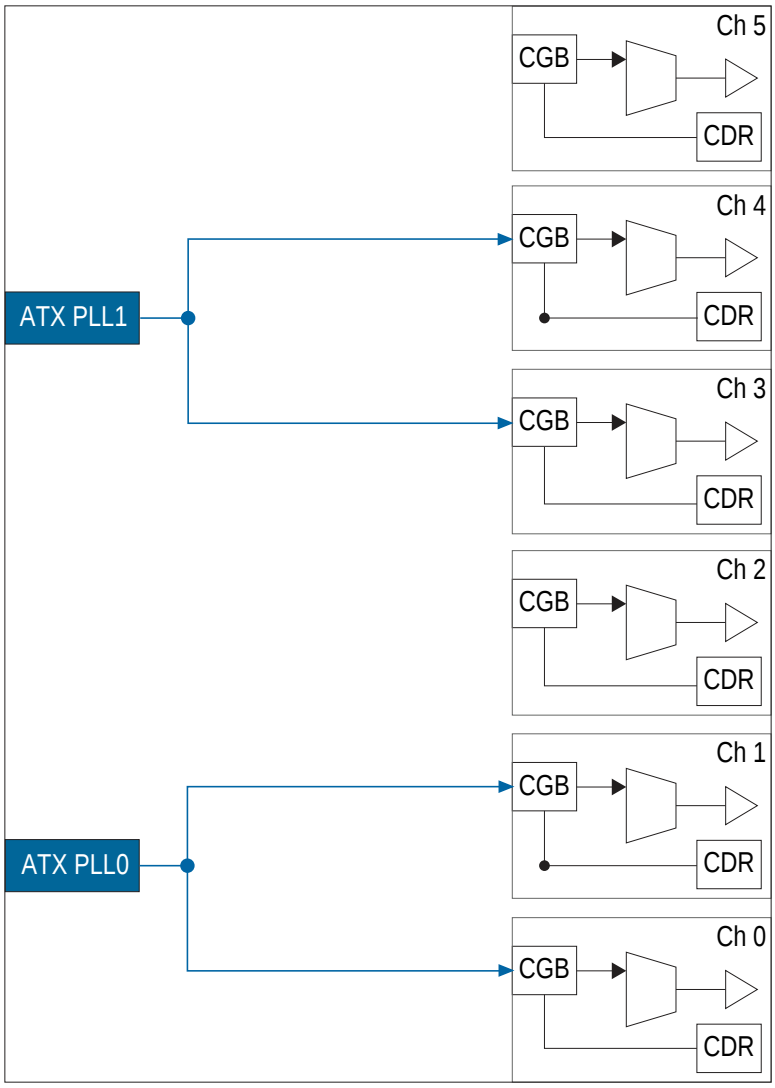
GT Clock Lines

GT clock lines are dedicated clock lines available only in Arria 10 GT devices.

Each ATX PLL has two dedicated GT clock lines which connect the PLL directly to the transceiver channels within a transceiver bank. The top ATX PLL drives channels 3 and 4, and the bottom ATX PLL drives channels 0 and 1. These connections bypass the rest of the clock network for higher performance. These channels can be used only for non-bonded configurations.

[29] The maximum data rate supported numbers are pending characterization data.

Figure 3-10: GT Clock Lines



Clock Generation Block

In Arria 10 devices, there are two types of clock generation blocks (CGBs)

- Local clock generation block (local CGB)
- Master clock generation block (master CGB)

Each transmitter channel has a local clock generation block (CGB). For non-bonded channel configurations, the serial clock generated by the transmit PLL drives the local CGB of each channel. The local CGB generates the parallel clock used by the serializer and the PCS.

There are two standalone master CGBs within each transceiver bank. The master CGB provides the same functionality as the local CGB within each transceiver channel. The output of the master CGB can be routed to other channels within a transceiver bank using the x6 clock lines. The output of the master CGB can also

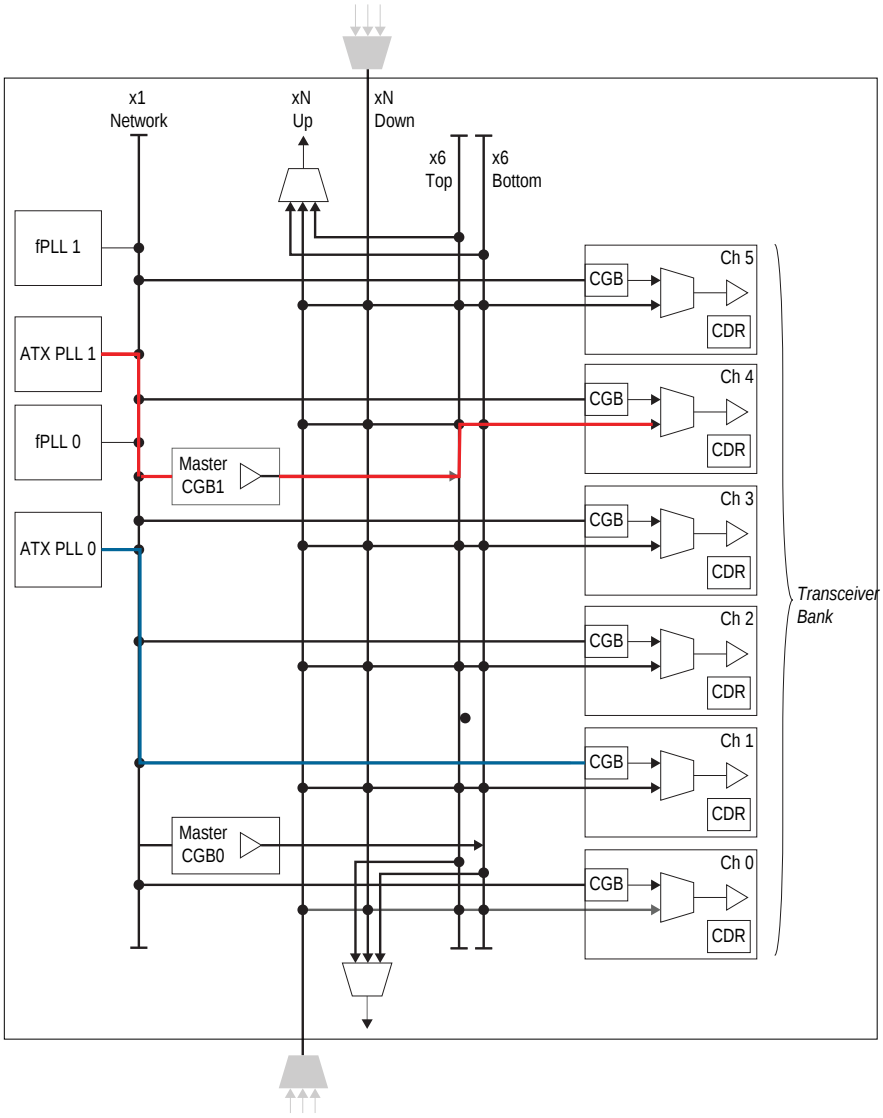
be routed to channels in other transceiver banks using the xN clock lines. Each transmitter channel has a multiplexer to select its clock source from either the local CGB or the master CGB.

Figure 3-11: Clock Generation Block and Clock Network

The local clock for each transceiver channel can be sourced from either the local CGB via the x1 network or the master CGB via the x6/xN network.

For example, as shown by the red highlighted path, the ATX PLL drives the x1 network which in turn drives the master CGB. The master CGB then drives the x6 clock network which routes the clocks to the local channels.

As shown by the blue highlighted path, the ATX PLL can also drive the x1 clock network which can directly feed a channel's local CGB. In this case, the low speed parallel clock is generated by the local CGB.



FPGA Fabric-Transceiver Interface Clocking

The FPGA fabric-transceiver interface consists of clock signals from the FPGA fabric into the transceiver and clock signals from the transceiver into the FPGA fabric. These clock signals use the global (GCLK), regional (RCLK), and periphery (PCLK) clock networks in the FPGA core.

The transmitter channel forwards a parallel output clock (`tx_clkout`) to the FPGA fabric to clock the transmitter data and control signals. The receiver channel forwards a parallel output clock to the FPGA fabric to clock the data and status signals from the receiver into the FPGA fabric. Based on the receiver channel configuration, the parallel output clock is recovered from either the receiver serial data or the`rx_clkout` clock (in configurations without the rate matcher) or the `tx_clkout` clock (in configurations with the rate matcher).

The divided versions of the `tx_clkout` and `rx_clkout` are available as `tx_pma_div_clkout` and `rx_pma_div_clkout` respectively.

The output frequency of `tx_pma_div_clkout` and `rx_pma_div_clkout` can be one of the following:

- A divided down version of the `tx_clkout` or `rx_clkout` respective where divide by 1 and divide by 2 ratios are available.
- A divided down version of the serializer clock where divide by 33, 40, and 66 ratios are available.

The `tx_pma_div_clkout` and `rx_pma_div_clkout` can be used as `tx_coreclkin` or `rx_coreclkin` and provide additional flexibility.

These clocks can be used to meet core timing by operating the TX and RX FIFO in double-width mode, as this halves the required clock frequency at the PCS to FPGA interface. These clocks can also be used to clock the core side of the TX and RX FIFOs when the Enhanced PCS Gearbox is used.

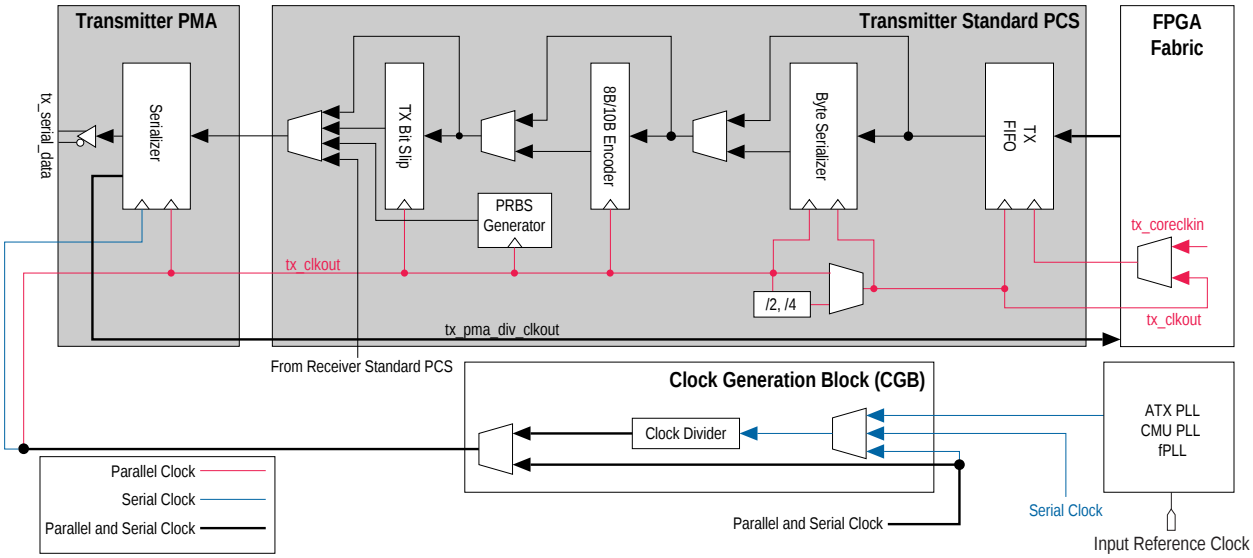
For example, if the Enhanced PCS Gearbox was engaged with a 66:40 ratio, the `tx_pma_div_clkout` could be used with a divide by 33 ratio to clock the write side of the TX FIFO, instead of using a PLL to generate the required clock frequency, or using an external clock source.

Transmitter Data Path Interface Clocking

The clocks generated by the PLLs are used to clock the channel PMA and PCS blocks. The clocking architecture is different for the standard PCS and the enhanced PCS.

Figure 3-12: Transmitter Standard PCS and PMA Clocking

The master/local CGB provides the high speed serial clock to the serializer of the transmitter PMA, and the low speed parallel clock to the transmitter PCS.



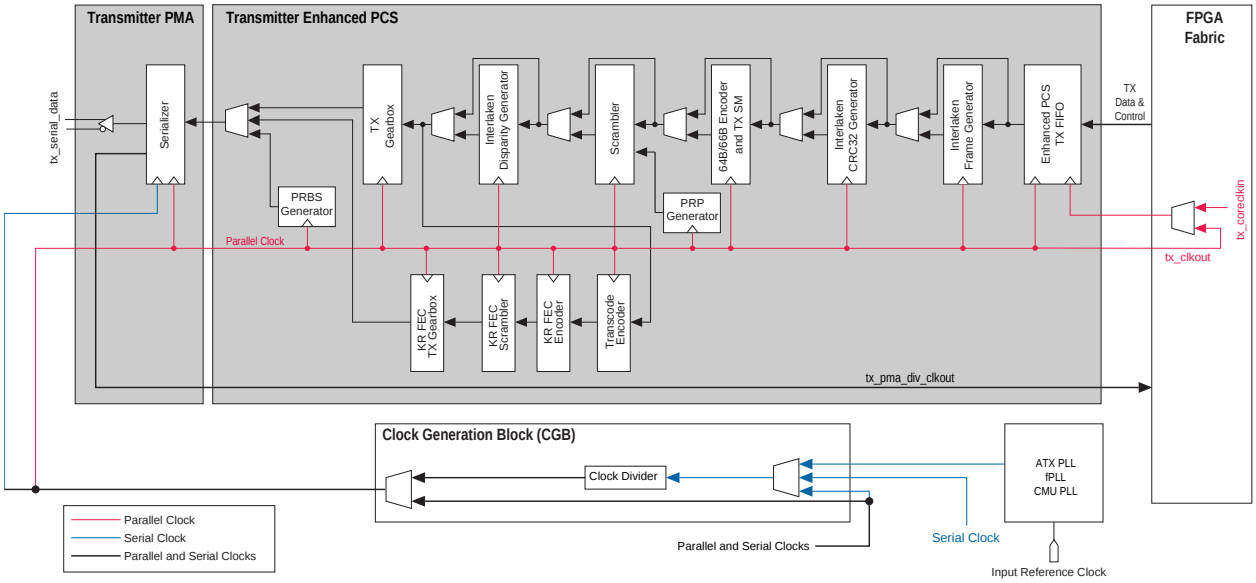
In the Standard PCS, for configurations that do not use the byte serializer, the parallel clock is used by all the blocks up to the read side of the TX phase compensation FIFO. For configurations that use the byte serializer block, the clock is divided by 2 for the byte serializer and the read side of the TX phase compensation FIFO. The clock used to clock the read side of the TX phase compensation FIFO is also forwarded to the FPGA fabric to provide an interface between the FPGA fabric and the transceiver.

If the `tx_clkout` that is forwarded to the FPGA fabric is used to clock the write side of the phase compensation FIFO, then both sides of the FIFO have 0 ppm frequency difference because it is the same clock which is used.

If you choose to use a different clock than the `tx_clkout` to clock the write side of the phase compensation FIFO, then you must ensure that the clock provided must have 0 ppm frequency difference with respect to the `tx_clkout`.

Figure 3-13: Transmitter Enhanced PCS and PMA Clocking

The Master/Local CGB provides the serial clock to the serializer of the transmitter PMA, and the parallel clock to the transmitter PCS.



In the Enhanced PCS, the parallel clock is used by all the blocks up to the read side of the TX phase compensation FIFO.

For the enhanced PCS, the transmitter PCS forwards the following clocks to the FPGA fabric:

- `tx_clkout` for each transmitter channel in non-bonded configuration
- `tx_clkout[0]` for all transmitter channels in bonded configuration

You can clock the transmitter datapath interface by using one of the following methods:

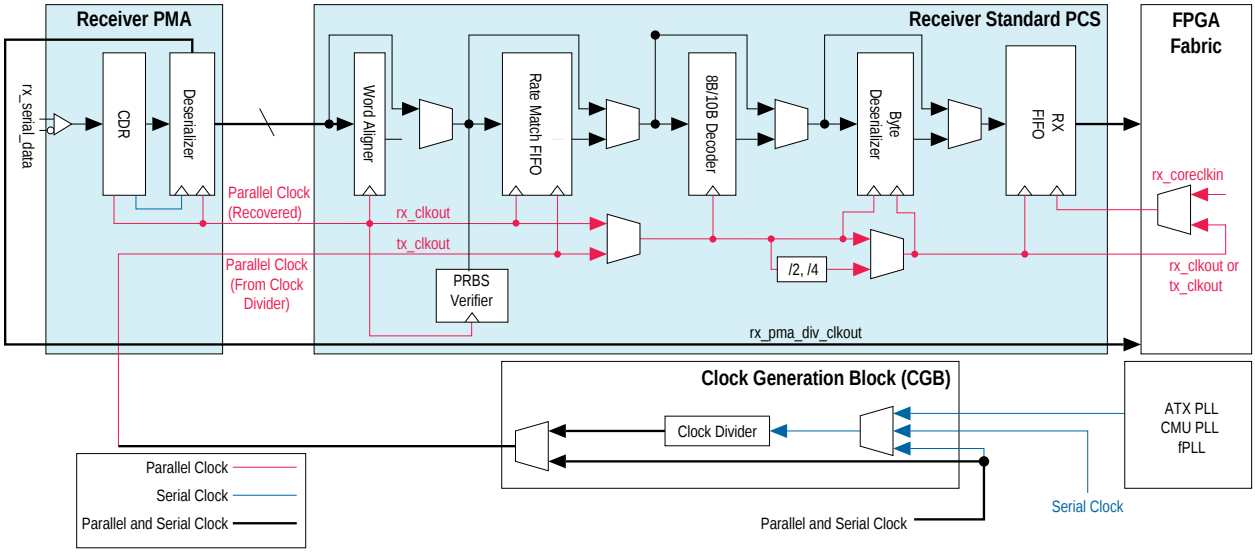
- Quartus II selected transmitter datapath interface clock
- User selected transmitter datapath interface clock

Receiver Data Path Interface Clocking

The CDR block present in the PMA of each channel recovers the serial clock from the incoming data. The CDR block also divides the recovered serial clock to generate the recovered parallel clock. Both the recovered serial and the recovered parallel clocks are used by the deserializer. The receiver PCS can use the following clocks based on the configuration of the receiver channel:

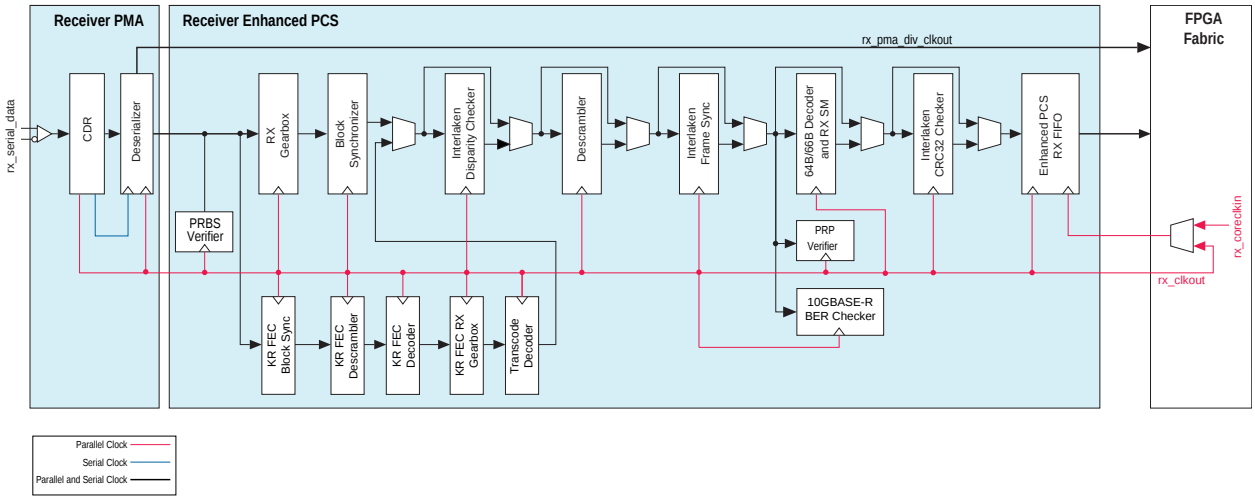
- Recovered parallel clock from the CDR in the PMA
- Parallel clock from the clock divider used by the transmitter PCS (if enabled) for that channel

Figure 3-14: Receiver Standard PCS and PMA Clocking



All configurations that use the standard PCS channel must have a 0 ppm phase difference between the receiver datapath interface clock and the read side clock of the RX phase compensation FIFO.

Figure 3-15: Receiver Enhanced PCS and PMA Clocking



The receiver PCS forwards the following clocks to the FPGA fabric:

- `rx_clkout` — for each receiver channel in a non-bonded configuration when the rate matcher is not used.
- `tx_clkout` — for each receiver channel in a non-bonded configuration when the rate matcher is used.
- single `rx_clkout[0]` — for all receiver channels in a bonded configuration (for both cases, when rate matcher is used and when rate matcher is not used)

You can clock the receiver datapath interface using either one of the following methods:

- Quartus II selected receiver datapath interface clock
- User-selected receiver datapath interface clock

Channel Bonding

For Arria 10 devices, two types of bonding modes are available:

- PMA bonding
- PMA and PCS bonding

PMA bonding refers to the minimization of skew across the PMA channels. Only the PMA portion of the transceiver datapath is skew compensated while the PCS is not skew compensated. x6 / xN clock networks or PLL feedback compensation are used for skew compensation in PMA bonding.

In PMA and PCS bonding, both the PMA and the PCS blocks of the transceiver datapath are skew compensated. For PCS bonding, a master PCS channel provides the synchronized and skew compensated control signals (such as the FIFO reset signals) to each of the slave channels and ensures minimum skew across the bonded channel group.

Skew Calculations

To calculate the maximum skew between the channels, the following parameters are used:

- PMA to PCS datapath interface width (S)
- Maximum difference in number of parallel clock cycles between deassertion of each channel's FIFO reset (N)

To calculate the channel skew, the following five scenarios are considered:

1. Non-bonded

   In this case, both the PMA and PCS are non-bonded. Skew ranges from 0 UI to [(S-1) + N*S] UI.

2. PMA bonding using x6 / xN clock network

   In this case, the PCS is non-bonded. Skew ranges from [0 to (N*S)] UI + x6/xN clock skew.

3. PMA bonding using the PLL feedback compensation clock network

   In this case, the PCS is non-bonded. Skew ranges from [0 to (N*S)] UI + (reference clock skew) + (x6 clock skew).

4. PMA and PCS bonding using the x6 / xN clock network

   Skew = x6/xN clock skew.

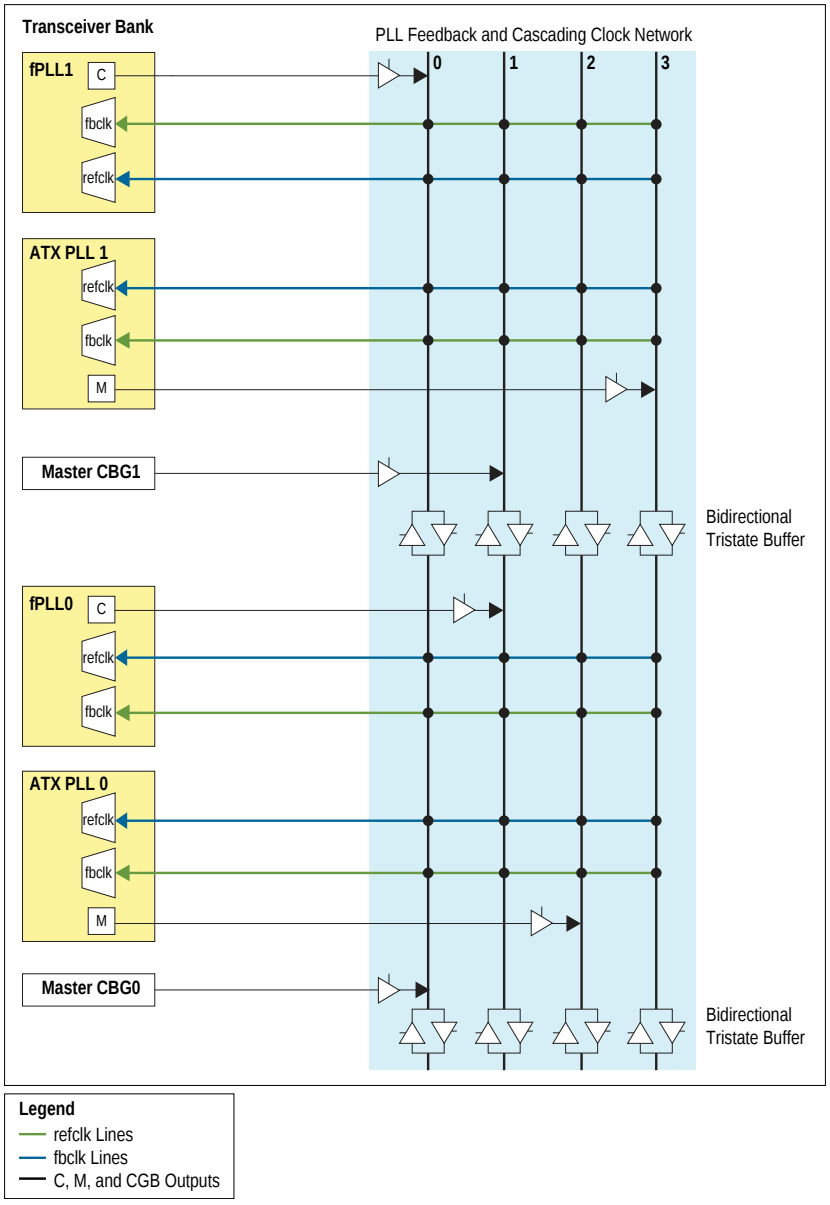
5. PMA and PCS bonding using PLL feedback compensation clock network

   Skew = (reference clock skew) + (x6 clock skew)

PLL Feedback and Cascading Clock Network

The PLL feedback and cascading clock network spans the entire side of the device, and is used for PLL feedback compensation bonding and PLL cascading.

Figure 3-16: PLL Feedback and Cascading Clock Network



To support PLL feedback compensation bonding and PLL cascading, the following connections are present:

1. The **divided clock output** (the C counter output for fPLL or M counter output for ATX PLL) of all PLLs drives the **feedback and cascading clock** network.
2. The **feedback and cascading clock** network drives the **feedback clock** input of all PLLs.
3. The **feedback and cascading clock** network drives the **reference clock** input of all PLLs.
4. The **master CGB's parallel clock output** drives the **feedback and cascading clock** network.

For PLL cascading, connections (1) and (3) are used to connect the output of one PLL to the reference clock input of another PLL. Arria 10 transceivers support only fPLL to fPLL or fPLL to ATX PLL cascading.

For PLL feedback compensation bonding, connections (2) and (4) are used to connect the master CGB's parallel clock output to the PLL feedback clock input port.

PLL feedback compensation bonding can be used instead of xN bonding. The primary difference between PLL feedback compensation and xN bonding configurations, is for PLL feedback compensation, the bonded interface is broken down into smaller groups of 6 bonded channels within a transceiver bank. A PLL within each transceiver bank (ATX PLL or fPLL) is used as a transmit PLL. All the transmit PLLs share the same input reference clock.

In xN bonding configurations, one PLL is used for each bonded group. Based on the number of channels in a bonded group, there are speed limitations for xN bonding. In PLL feedback compensation, one PLL is used for each transceiver bank that the bonded group spans. However, there are no speed limitations in PLL feedback compensation bonding, other than the natural speed limitations of the transceiver channel and the PLL.

For feedback compensation bonding, the low speed parallel clock must be the same frequency as the reference clock for the PLL.

Using PLLs and Clock Networks

In Arria 10 devices, PLLs are not integrated in the Native PHY IP. You must instantiated by the user through a separate PLL IP. PLL merging is no longer performed by Quartus II software. This gives you more control, transparency, and flexibility in the design process. You can specify the channel configuration and PLL usage.

Related Information

Clock Networks

Non-bonded configurations

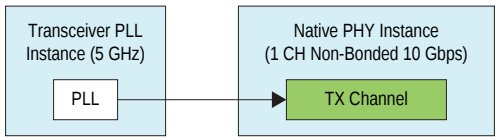
In a non-bonded configuration, only the high speed serial clock is routed from the transmitter PLL to the transmitter channel. The low speed parallel clock is generated by the local clock generation block (CGB) present in the transceiver channel. For non-bonded configurations, because the channels are not related to each other and the feedback path is local to the PLL, the skew between channels cannot be calculated. Also, the skew introduced by the clock network is not compensated.

Single Channel x1 Non-Bonded Configuration

In x1 non-bonded configuration, the PLL source is local to the transceiver bank and the x1 clock network is used to distribute the clock from the PLL to the transmitter channel.

For a single channel design, a PLL is used to provide the clock to a transceiver channel.

Figure 3-17: PHY IP and PLL IP Connection for Single Channel x1 Non-Bonded Configuration Example



To implement this configuration, instantiate a PLL IP and a PHY IP and connect them together as shown in the above figure.

Steps to implement a Single Channel x1 Non-Bonded Configuration

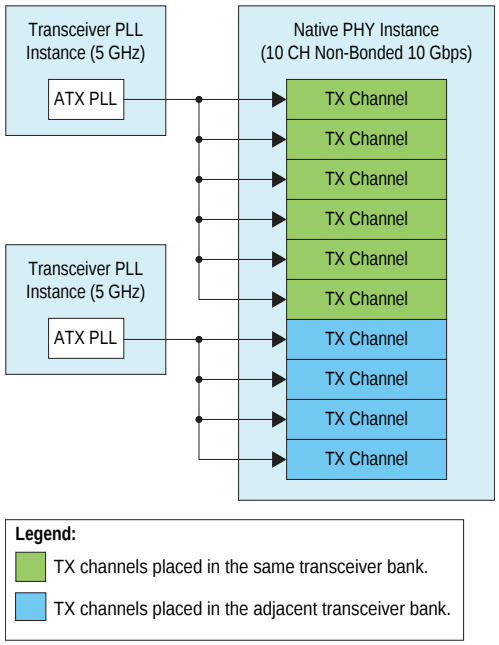
1. Choose the PLL IP (ATX PLL, fPLL, or CMU PLL) you want to instantiate in your design and instantiate the PLL IP.
   - Refer to **Instantiating ATX PLL** on page 3-5 or **Instantiating CMU PLL IP** on page 3-21 or **Instantiating fPLL IP** on page 3-13 for detailed steps.
2. Configure the PLL IP using the **MegaWizard Plug-In Manager**
   - For ATX PLL IP, set the PLL feedback type to **internal** and do not include the Master CGB.
   - For fPLL IP, set the PLL feedback operation mode to **direct.**
   - For CMU PLL IP, specify the reference clock and the data rate. No special configuration rule is required.
3. Configure the Native PHY IP using the **MegaWizard Plug-In Manager**
   - Set the **Native PHY IP TX Channel bonding mode** to **Non Bonded**.
4. Connectivity
   - Connect the PLL IP to Native PHY IP.
   - Connect the `tx_serial_clk` output port of the PLL to IP to the corresponding `tx_serial_clk0` input port of the Native PHY IP. This port represents the input to the local CGB of the channel. The `tx_serial_clk` for the PLL represents the high speed serial clock generated by the PLL.

Multi-Channel x1 Non-Bonded Configuration

This configuration is an extension of the x1 non-bonded case. In the following example, 10 channels are connected to two instances of the PLL IP. Two PLL instances are required because PLLs using the x1 clock network can only span the 6 channels within the same transceiver bank. A second PLL instance is required to provide the clock to the remaining 4 channels.

Because 10 channels are not bonded and are unrelated, you can use a different PLL type for the second PLL instance. It is also possible to use more than two PLL IPs and have different PLLs driving different channels. If some channels are running at different data rates, then you need different PLLs driving different channels.

Figure 3-18: PHY IP and PLL IP Connection for Multi-Channel x1 Non-Bonded Configuration



Steps to implement a Multi-Channel x1 Non-Bonded Configuration

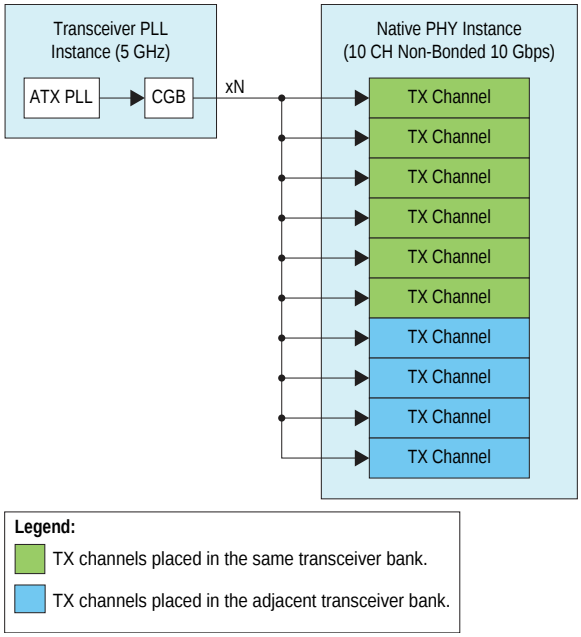
1. Choose the PLL IP (ATX PLL, fPLL, or CMU PLL) you want to instantiate in your design and instantiate the PLL IP.
   - Refer to **Instantiating ATX PLL** on page 3-5 or **Instantiating CMU PLL IP** on page 3-21 or **Instantiating fPLL IP** on page 3-13 for detailed steps.
2. Configure the PLL IP using the **MegaWizard Plug-In Manager**
   - For ATX PLL IP, set the PLL feedback type to **internal** and do not include the Master CGB.
   - For fPLL IP, set the PLL feedback operation mode to **direct.**
   - For CMU PLL IP, specify the reference clock and the data rate. No special configuration rule is required.
3. Configure the Native PHY IP using the **MegaWizard Plug-In Manager**
   - Set the **Native PHY IP TX Channel bonding mode** to **Non-Bonded**.
   - Set the number of channels as per your design requirement. In this example, the number of channels is set to 10.
4. Connectivity
   - Create a top level wrapper to connect the PLL IP to Native PHY IP.
   - The `tx_serial_clk output` port of the PLL IP represents the high speed serial clock.
   - The Native PHY IP has 10 (for this example) `tx_serial_clk input` ports. Each port corresponds to the input of the local CGB of the transceiver channel.
   - As shown in the figure above, connect the first 6 `tx_serial_clk input` to the first transceiver PLL instance.
   - Connect the remaining 4 `tx_serial_clk input` to the second transceiver PLL instance.

Multi-Channel xN Non-Bonded Configuration

Using the xN non-bonded configuration reduces the number of PLL resources and the reference clock sources used.

Figure 3-19: PHY IP and PLL IP Connection for Multi-Channel xN Non-Bonded Configuration

In this example, the same PLL is used to drive 10 channels across two transceiver banks.



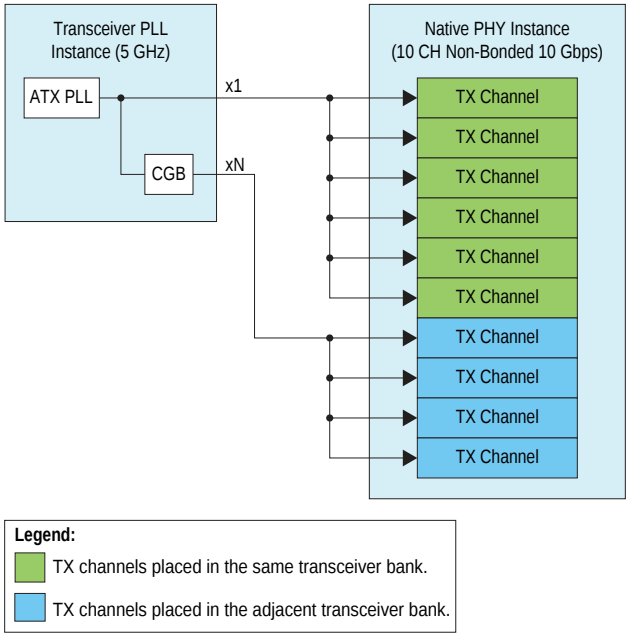
Steps to implement a multi-channel xN non-bonded configuration

1. You can use either the ATX PLL or fPLL for multi-channel xN non-bonded configuration.
   - Refer to **Instantiating ATX PLL** on page 3-5 or **Instantiating fPLL IP** on page 3-13 for detailed steps.
2. Configure the PLL IP using the **MegaWizard Plug-In Manager**
   - Because the CMU PLL cannot drive the master CGB, only the ATX PLL or fPLL can be used for this example.
   - Enable **Include Master Clock Generation Block** and turn ON **Enable high frequency serial clock output port**.
3. Configure the Native PHY IP using the **MegaWizard Plug-In Manager**
   - Set the **Native PHY IP TX Channel bonding mode** to **Non-Bonded**.
   - Set the number of channels as per your design requirement. In this example, the number of channels is set to 10.
4. Connectivity

- Create a top level wrapper to connect the PLL IP to Native PHY IP.
- In this case, the PLL IP has `mcgb_serial_clk` output port. This represents the xN clock line.
- The Native PHY IP has 10 (for this example) `tx_serial_clk input` ports. Each port corresponds to the input of the local CGB of the transceiver channel.
- As shown in the figure above, connect the `mcgb_serial_clk` output port of the PLL IP to the 10 `tx_serial_clk input` ports of the Native PHY IP.

Figure 3-20: Multi-Channel x1/xN Non-Bonded Example

PLL IP has a `tx_serial_clk` output port. This port can optionally be used to clock the six channels within the same transceiver bank as the PLL. These channels are clocked by the x1 network. The remaining four channels outside the transceiver bank are clocked by the xN clock network. This is shown in the following figure.



Bonded configurations

In a bonded configuration, both the high speed serial and low speed parallel clocks are routed from the transmitter PLL to the transmitter channel. In this case, the local CGB in each channel is bypassed and the parallel clocks generated by the master CGB are used to clock the network.

In bonded configurations, the transceiver clock skew between the channels is minimum. Bonded configurations are used for channel bonding in protocols such as PCIe and XAUI.

There are two methods for channel bonding in Arria 10 devices:

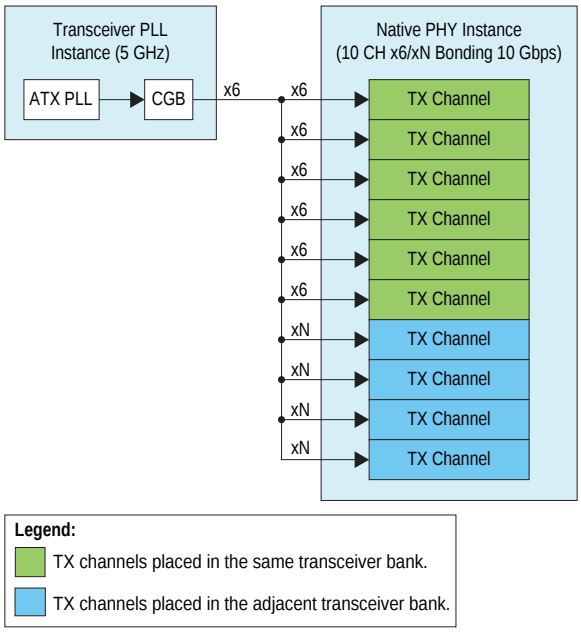
- x6/xN bonding
- PLL feedback compensation

x6/xN Bonding Mode

In x6/xN bonding mode, the same PLL drives multiple channels spanning more than one transceiver bank. The steps below explain the x6/xN bonding process:

1. The ATX PLL or the fPLL generates a high speed serial clock.
2. The PLL drives the high speed serial clock to the Master CGB via the x1 clock network.
3. The Master CGB drives the high speed serial and the low speed parallel clock into the x6 clock network.
4. The x6 clock network feeds the TX clock mux for the transceiver channels within the same transceiver bank. The local CGB in each transceiver channel is bypassed.
5. To drive the channels in adjacent transceiver banks, the x6 clock network drives the xN clock network.

Figure 3-21: PHY IP and PLL IP Connection for x6/xN Bonding Mode



The advantage of the xN bonding mode is that it uses the least number of resources. One PLL and one master CGB can drive all the channels.

A disadvantage of the xN bonding mode is the data rate restriction. Because a centrally generated set of high speed serial and low speed parallel clocks are distributed to all TX channels, there is a limit on the maximum data rate obtained. This maximum data rate depends on the clock network span.

Steps to implement a x6/xN bonded configuration

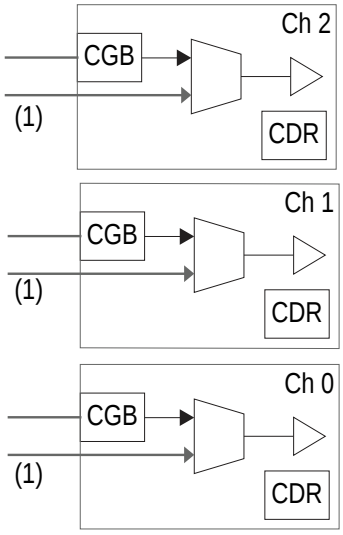
1. You can instantiate either the ATX PLL of the fPLL for x6/xN bonded configuration.
   - Refer to **Instantiating ATX PLL** on page 3-5 or **Instantiating fPLL IP** on page 3-13 for detailed steps.
2. Configure the PLL IP using the **MegaWizard Plug-In Manager**

- Because the CMU PLL cannot drive the Master CGB, only the ATX PLL or fPLL can be used for this example.
- Enable **Include Master Clock Generation Block** and turn ON **Enable high frequency serial clock output port**.

3. Configure the Native PHY IP using the **MegaWizard Plug-In Manager**
   - Set the **Native PHY IP TX Channel bonding mode** to either **PMA bonding** or **PMA/PCS bonding**.
   - Set the number of channels required by your design. In this example, the number of channels is set to 10.
4. Connectivity
   - Create a top level wrapper to connect the PLL IP to Native PHY IP.
   - In this case, the PLL IP has `tx_bonding_clocks` output bus with width [5:0].
   - The Native PHY IP has `tx_bonding_clocks` input bus with width [5:0] multiplied by the number of transceiver channels (10 in this case). For 10 channels, the bus width will be [50:0].
   - Connect the PLL to the PHY IP by duplicating the output of the PLL[5:0] for the number of channels. For 10 channels, the Verilog syntax for the input port connection is `.tx_bonding_clocks ({number_of_channels{tx_bonding_clocks_output}})`.

Note: Although the above diagram looks similar to the 10-channel non-bonded configuration example, the clock input ports on the transceiver channels bypass the local CGB in x6/xN bonding configuration. This internal connection is taken care of when the **Native PHY channel bonding mode** is set to **Bonded**.

Figure 3-22: x6/xN Bonding Mode —Internal Channel Connections



Note: (1) The local CGB is bypassed by the clock input ports in bonded mode.

Related Information

xN Clock Lines on page 3-29
Information on xN Clock Network Span.

PLL Feedback Compensation Bonding Mode

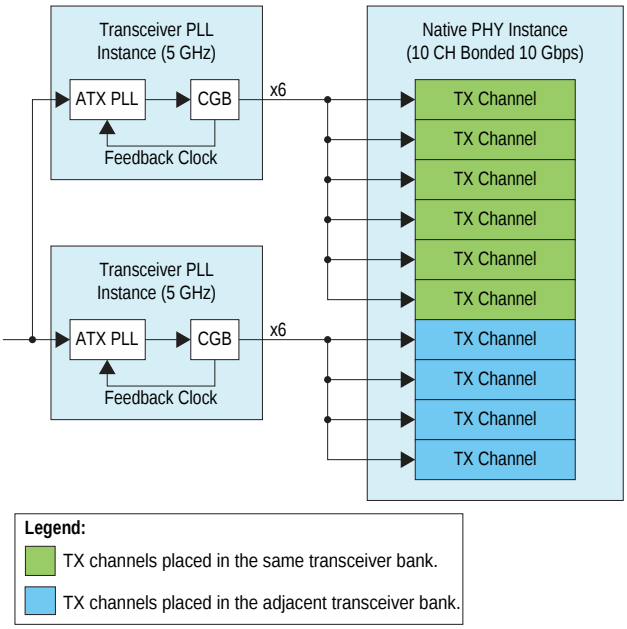
In this bonding mode, the data rate limitations of xN bonding mode are removed. This is achieved by dividing all channels into multiple bonding groups.

Each bonding group is driven by its own set of high speed serial and low speed parallel clocks. Each bonding group has its own PLL and master CGB. The PLL and the master CGB for different groups share the same reference clock in order to maintain the same phase relationship.

The following steps explain the PLL feedback compensation bonding process:

1. The same input reference clock drives the local PLL in each triplet or a 6-channel transceiver bank.
2. The local PLL for the bonding group drives the master CGB.
3. The master CGB feeds the x6 clock line. Master CGB drives the transceiver channels in the bonding group via the x6 clock network.
4. The parallel output of the master CGB is the feedback input to the PLL.
5. All channels are phase aligned to the same input reference clock.

Figure 3-23: PHY IP and PLL IP Connection for PLL Feedback Compensation Bonding



The data rate is limited by the x6 network speed limit. A disadvantage of using PLL feedback compensation bonding is that it consumes more PLL resources. Each transceiver bank consumes one PLL and one master CGB.

In PLL feedback compensation bonding mode, the N counter (reference clock divider) is bypassed in order to ensure that the reference clock skew is minimized between the PLLs in the bonded group. Because the N counter is bypassed the PLL reference clock has a fixed value for any given data rate.

The PLL IP **MegaWizard Plug-In Manager** GUI displays the required data rate in the **PLL reference clock frequency** drop down menu.

Steps to implement a PLL Feedback Compensation Bonding Configuration

1. Choose the PLL IP (ATX PLL or fPLL) you want to instantiate in your design and instantiate the PLL IP. Refer to **Instantiating ATX PLL** on page 3-5 or **Instantiating fPLL IP** on page 3-13 for detailed steps.
2. Configure the PLL IP using the **MegaWizard Plug-In Manager**
   - Because the CMU PLL cannot drive the master CGB, only the ATX PLL or fPLL can be used for this example.
   - If ATX PLL is used, set the following configuration settings:
     - Under the **PLL Tab**
       - Set the **PLL Feedback type** to **External**
       - Turn ON **Enable external feedback input port**.
     - Under the **Master Clock Generation Block Tab**
       - Enable **Include Master Clock Generation Block**
       - Turn ON **Enable Bonding Clock output ports.**
       - Turn ON **Enable feedback compensation bonding**
   - If fPLL is used, set the following configuration settings:
     - Under the **PLL Tab**
       - Set the **PLL Feedback type** to **Normal**
     - Under the **Master Clock Generation Block Tab**
       - Turn ON **Enable Bonding Clock output ports.**
3. Configure the Native PHY IP using the **MegaWizard Plug-In Manager**
   - Set the **Native PHY IP TX Channel bonding mode** to either **PMA bonding** or **PMA/PCS bonding**.
4. Connectivity
   - Create a top level wrapper to connect the PLL IP to Native PHY IP.
   - In this case, the PLL IP has `tx_bonding_clocks` output bus with width [5:0].
   - The Native PHY IP has `tx_bonding_clocks` input bus with width [5:0] multiplied by the number of channels in a transceiver bank. (six channels in the transceiver bank).
   - Unlike the x6/xN bonding mode, for this mode, the PLL should be instantiated multiple times. (1 PLL is required for each transceiver bank that is a part of the bonded group.) Instantiate a PLL for each transceiver bank used.
   - Connect the `tx_bonding_clocks` output from each PLL to (up to) six channels in the same transceiver bank.
   - Connect the PLL to the PHY IP by duplicating the output of the PLL[5:0] for the number of transceiver channels used in the bonding group.

Note: For this 10-channel example, two ATX PLLs are instantiated. 6 channels of the `tx_bonding_clocks` on the Native PHY IP are connected to the first ATX PLL and the remaining four channels are connected to the second ATX PLL's `tx_bonding_clock` outputs.

PLL cascading

In PLL cascading, the output of the first PLL feeds the input reference clock to the second PLL.

For example, if the input reference clock has a fixed frequency, and the desired data rate was not an integer multiple of the input reference clock. In this case, the first PLL can be used to generate the correct reference clock frequency. This output is fed as the input reference clock to the second PLL. This second PLL generates the clock frequency required for the desired data rate.

The transceivers in Arria 10 devices support fPLL to fPLL or fPLL to ATX PLL cascading. The first PLL must be a fPLL.

The following example describes fPLL to ATX PLL cascading.

PLL IP MegaWizard configuration

1. Generate the fPLL IP. The first PLL used in a PLL cascading configuration must be a fPLL.
2. Turn ON **Enable Cascade clock output port (fPLL to ATX PLL cascading)**.
3. Turn ON **Use as Core PLL** option
4. Configure the **number of clocks desired** and the **desired output clock parameters**. A message appears indicating which output clock port should be used as the cascading source.
5. Generate the ATX PLL IP (the second PLL in the design).
6. Configure the ATX PLL IP for the desired data rate and the reference clock frequency. Ensure that the reference clock frequency for the ATX PLL matches the output of the fPLL.

Note: No special configuration is required for the Native PHY instance.

Connectivity

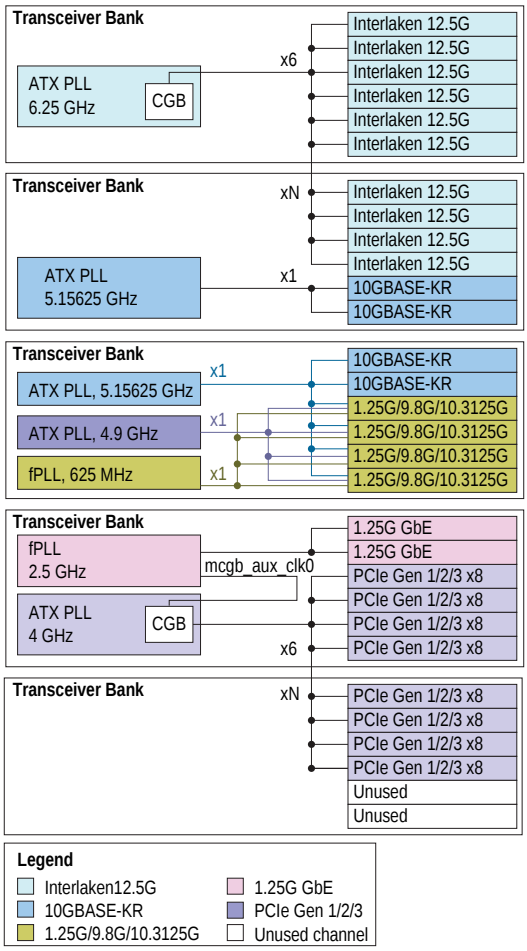
1. Instantiate both the fPLL and the ATX PLL.
2. The fPLL has an output port `atx_pll_cascade_clk`. Connect this port to the ATX PLL's `pll_refclk0` port.
3. For `pll_powerdown`, both the PLLs can share the same `pll_powerdown` or use independent power down sources.
4. OR the `pll_powerdown` of ATX PLL with `pll_lock` signal of the first PLL. This ensures that the second PLL (downstream PLL) is powered down until the first PLL has successfully locked to the input reference clock. This prevents the second PLL from trying to lock until the output of the first PLL is stable.

Note: The procedure for fPLL to fPLL cascading is similar to the one described here (fPLL to ATX PLL cascading).

Mix and Match Example

The Arria 10 transceiver architecture and Native PHY IP / PLL IP scheme allows great flexibility. It is easy to share PLLs and reconfigure data rates. The following design example illustrates PLL sharing and both bonded and non-bonded clocking configurations.

Figure 3-24: Mix and Match Design Example



PLL Instances

In this example, five ATX PLL instances and two fPLL instances are used. Choose an appropriate reference clock for each PLL instance. The **MegaWizard Plug-In Manager** GUI lists the available choices.

Use the following data rates and configuration settings for PLL IPs:

- Transceiver PLL Instance 0: ATX PLL with output clock frequency of 6.25 GHz
  - Enable the Master CGB and bonding output clocks.
- Transceiver PLL instance 1: ATX PLL with output clock frequency of 5.1625 GHz
- Transceiver PLL instance 2: ATX PLL with output clock frequency of 5.1625 GHz
- Transceiver PLL instance 3: ATX PLL with output clock frequency of 4.9 GHz

- Transceiver PLL instance 4: fPLL with output clock frequency of 0.625 GHz
  - Select the **Use as Transceiver PLL** option.
- Transceiver PLL instance 5: fPLL with output clock frequency of 2.5 GHz
  - Select **Enable PCIe clock output port** option
  - Select **Use as Transceiver PLL** option
    - Set Protocol Mode to PCIe Gen2
  - Select the **Use as Core PLL** option
    - Set the **Desired frequency** to 500 MHz with a phase shift of 0 ps.
- Transceiver PLL instance 6: ATX PLL with output clock frequency of 4 GHz
  - Enable Master CGB and bonding output clocks
  - Select **Enable PCIe clock switch interface** option
  - Set **Number of Auxiliary MCGB Clock Input ports** to 1

Native PHY IP Instances

In this example, four Transceiver Native PHY IP instances and four 10GBASE-KR PHY IP instances are used. Use the following data rates and configuration settings for the PHY IPs:

- 12.5 Gbps Interlaken with a bonded group of 10 channels
  - Set the Interlaken 10x12.5 Gbps preset from the Arria 10 Transceiver Native PHY IP GUI.
  - Refer to **Interlaken** on page 2-62 for more details.
- Custom multi-data rate 1.25G/9.8G/10.3125 Gbps non-bonded group of four channels
  - Set the **Number of data channels** to 4.
  - Set **TX channel bonding** to Not Bonded.
  - Under the **TX PMA** tab, set the **Number of TX PLL clock inputs per channel** to 3.
  - Under the **RX PMA** tab, set the **Number of CDR reference clocks** to 3.
- 1.25 Gbps Gigabit Ethernet with a non-bonded group of two channels
  - Set the **GIGE-1.25Gbps** preset from the Arria 10 Transceiver Native PHY IP GUI.
  - Change the **Number of data channels** to 2.
- PCIe Gen3 with a bonded group of 8 channels
  - Set the **PCIe PIPE Gen3x8** preset from the Arria 10 Transceiver Native PHY IP GUI.
  - Under **TX Bonding options**, set the **PCS TX channel bonding master** to channel 5

    Note: The PCS TX channel bonding master must be physically placed in channel 1 or channel 4 within a transceiver bank. In this example, the 5th channel of the bonded group is physically placed at channel 1 in the transceiver bank.

  - Refer to **PCI Express** on page 2-179 for more details.
- 10.3125 Gbps 10GBASE-KR non-bonded group of 4 channels
  - Instantiate the Arria 10 1G/10GbE and 10GBASE-KR PHY IP four times, with one instance for each channel.
  - Refer to **10GBASE-KR PHY IP with FEC Option** on page 2-103 for more details.

Connection Guidelines for PLL and Clock Networks

- For 12.5 Gbps Interlaken with a bonded group of 10 channels, connect the `tx_bonding_clocks` to the transceiver PLL's `tx_bonding_clocks` output port. Make this connection for all 10 bonded channels. This connection uses a master CGB and the x6 / xN clock line to reach all the channels in the bonded group.
- Connect the `tx_serial_clk` port of the first two instances of the 10GBASE-KR PHY IP to the `tx_serial_clk` port of PLL instance 1 (ATX PLL at 5.1625 GHz). This connection uses the x1 clock line within the transceiver bank.
- Connect the `tx_serial_clk` port of the remaining two instances of the 10GBASE-KR PHY IP to the `tx_serial_clk` port of the PLL instance 2 (ATX PLL at 5.1625 GHz). This connection uses the x1 clock line within the transceiver bank.
- Connect the three `tx_serial_clk` ports for the custom multi-data rate PHY IP as follows:
  - Connect `tx_serial_clk0` port to the `tx_serial_clk` port of PLL instance 2 (ATX PLL at 5.1625 GHz). This PLL instance is shared with the two 10GBASE-KR PHY IP channels and also uses the x1 clock line within the transceiver bank.
  - Connect the `tx_serial_clk1` port to the `tx_serial_clk` port of the PLL instance 3 (ATX PLL at 4.9 GHz). This connection uses the x1 clock line within the transceiver bank.
  - Connect the `tx_serial_clk2` port to the `tx_serial_clk` port of the PLL instance 4 (ATX PLL at 4.9 GHz). This connection uses the x1 clock line within the transceiver bank.
- Connect the 1.25 Gbps Gigabit Ethernet non-bonded PHY IP instance to the `tx_serial_clk` port of the PLL instance 5. Make this connection twice, one for each channel. This connection uses the x1 clock line within the transceiver bank.
- Connect the PCIe Gen3 bonded group of 8 channels as follows:
  - Connect the `tx_bonding_clocks` of the PHY IP to the `tx_bonding_clocks` port of the Transceiver PLL Instance 6. Make this connection for each of the 8 bonded channels.
  - Connect the `pipe_sw_done` of the PHY IP to the `pipe_sw` port of the transceiver PLL instance 6.
  - Connect the `pll_pcie_clk` port of the PLL instance 5 to the PHY IP's `pipe_hclk_in` port.
  - Connect `tx_serial_clk` port of the PLL instance 5 to the `mcgb_aux_clk0` port of the PLL instance 6. This connection is required as a part of the PCIe speed negotiation protocol.

Resetting Transceiver Channels

4

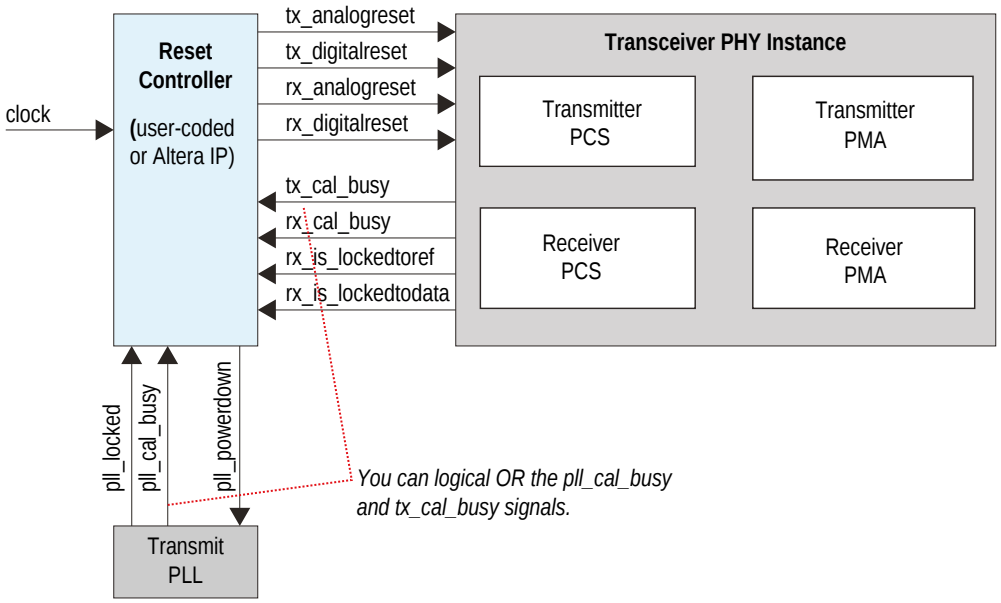
2013.12.02

UG-A10XCVR

Subscribe Send Feedback

To ensure that transceiver channels are ready to transmit and receive data, you must properly reset the transceiver PHY. The recommended reset sequence ensures that the physical coding sublayer (PCS) and physical medium attachment (PMA) in each transceiver channel initialize and function correctly.

Figure 4-1: Typical Transceiver PHY Implementation



© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

When Is Reset Required?

You can reset the transmitter (TX) and receiver (RX) data paths independently or together. The recommended reset sequence requires reset and initialization of the PLL driving the TX or RX channels, as well as the TX and RX datapaths. A reset may be required after any of the following events:

Table 4-1: Reset Conditions

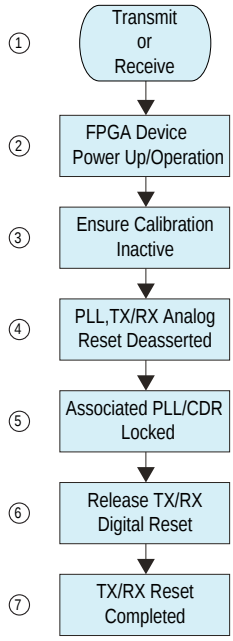
Event	Reset Requirement
Device power up and configuration	Requires reset to the transceiver PHY and the associated PLLs to a known initialize state.
PLL reconfiguration	Requires reset to ensure that the PLL acquires lock at optimal operating conditions and also to reset the PHY.
PLL reference clock frequency change	Requires reset to the PLL to ensure PLL lock. You must also reset the PHY.
PLL recalibration	Requires reset to the PLL to ensure PLL lock. You must also reset the PHY.
PLL lock loss or recovery	Requires reset after a PLL acquired lock from a momentary loss of lock. You must also reset the PHY.
Channel dynamic reconfiguration	Requires reset to the PLL and the PHY to initialize blocks for the new configuration.
Optical module connection	Requires reset of RX to ensure lock of incoming data.
RX CDR lock mode change	Requires reset of the RX channel any time the RX clock and data recovery (CDR) block switches from lock-to-reference to lock-to-data RX channel.

How Do I Reset?

You reset a transceiver PHY or PLL by integrating a reset controller in your system design to initialize the PCS and PMA blocks. You can save time by using the Altera-provided Transceiver PHY Reset Controller IP core, or you can implement your own reset controller that follows the recommended reset sequence. You can design your own reset controller if you require individual control of each signal for reset or need additional control or status signals as part of the reset functionality.

Recommended Reset Sequence

Figure 4-2: Transmitter and Receiver Reset Sequence



Resetting the Transmitter After Device Power-Up

The FPGA automatically calibrates the PLL at every power-up before entering user-mode. Perform a reset sequence after the device enters user-mode. Your User-Coded Reset Controller must comply with the reset sequence below to ensure a reliable transmitter initialization after the initial power-up calibration.

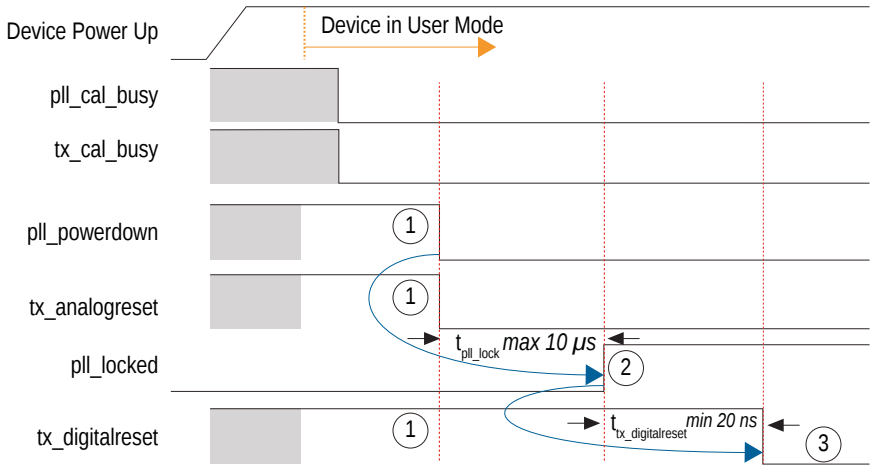
The following steps detail the transmitter reset sequence during device power-up. The step numbers correspond to the numbers in the waveform.

1. After the transmitter PLL `pll_powerdown` is deasserted, the `pll_locked` status is asserted after t_{pll_lock}.
2. Deassert `tx_digitalreset` after $t_{tx\_digitalreset}$.

   The transmitter is now out of reset and ready for operation.

Note: During calibration, `pll_locked` might assert and deassert as the calibration IP runs. After calibration, `pll_locked` is a steady signal. ATX PLL ref clock must be stable before `pll_powerdown` signal is deasserted .

Figure 4-3: Transmitter Reset Sequence After Power-Up



Note:

1. The TX PLL reference clock must be stable before the `pll_powerdown` signal is deasserted.

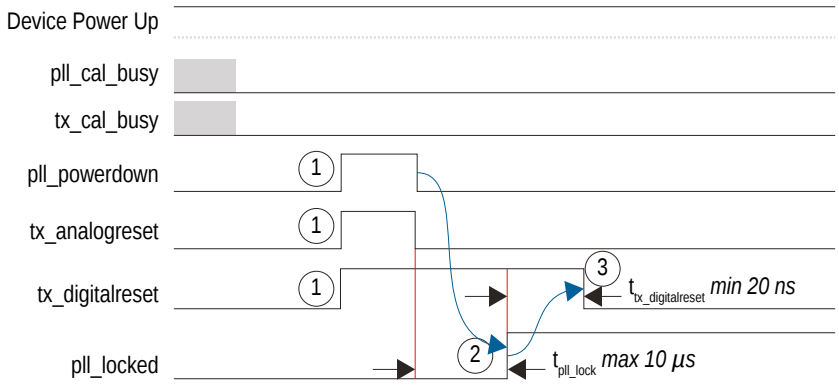
Resetting the Transmitter During Device Operation

Follow this reset sequence to reset the PLL or the analog or digital blocks of the transmitter at any point during the device operation. Use this reset to reestablish a link or after dynamic reconfiguration. The following steps detail the transmitter reset sequence during device operation. The step numbers correspond to the numbers in the waveform.

1. Perform the following steps:
   a. Assert `pll_powerdown`, `tx_analogreset` and `tx_digitalreset` while `pll_cal_busy` and `tx_cal_busy` are low.
   b. Hold `pll_powerdown` asserted for a minimum duration of t_{pll}.
   c. Deassert `tx_analogreset` at the same time or after you deassert `pll_powerdown`.
2. The `pll_locked` status signal goes high after TX PLL acquires lock.
3. Deassert `tx_digitalreset` after `pll_locked` goes high.

Note: You must reset the PCS blocks by asserting `tx_digitalreset`, every time you assert `pll_powerdown_powerdown` and `tx_analogreset`.

Figure 4-4: Transmitter Reset Sequence During Device Operation



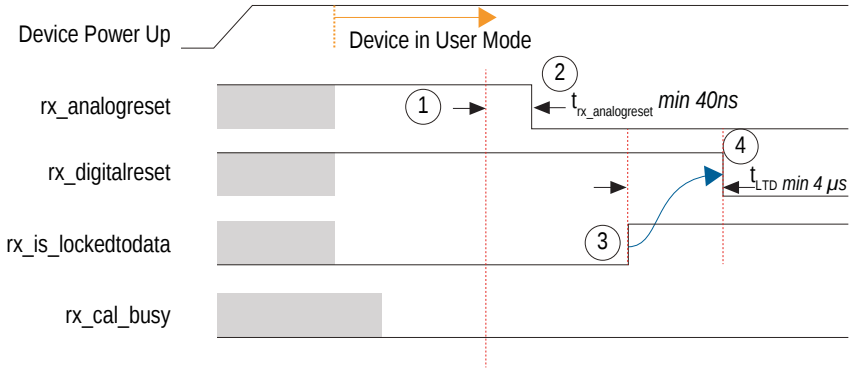
Resetting the Receiver After Device Power-Up

Follow this reset sequence to ensure a reliable receiver initialization after initial power-up. The following steps detail the receiver reset sequence after device configuration. The step numbers correspond to the numbers in the waveform.

1. Hold `rx_analogreset` and `rx_digitalreset` active at power-up to hold the receiver in reset.
2. Make sure the `rx_cal_busy` status is deasserted. Deassert `rx_analogreset` for a minimum duration of $t_{rx\_analogreset}$ after device enters user-mode. The `CONF_DONE` pin is asserted when the device enters user-mode.
3. Wait for `rx_is_lockedtodata` to go high.
4. Deassert `rx_digitalreset` after `rx_is_lockedtodata` is asserted for a minimum duration of t_{LTD}. If `rx_is_lockedtodata` is asserted and toggles, you must wait another additional t_{LTD} duration before deasserting `rx_digitalreset`.

   The receiver is now out of reset and ready for operation.

Figure 4-5: Receiver Reset Sequence Following Power-Up



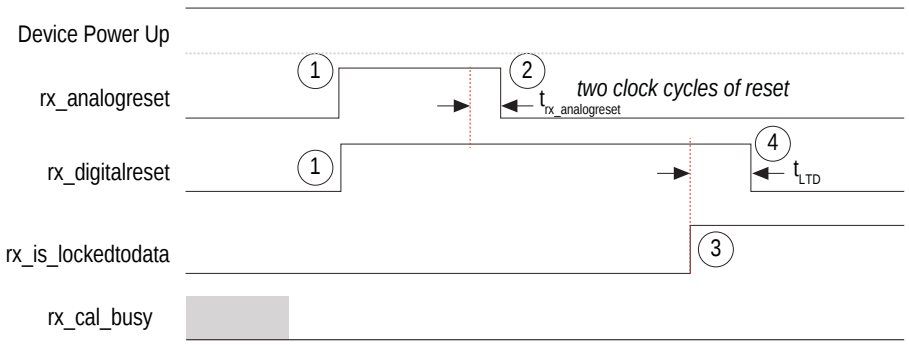
Note: `rx_is_lockedtodata` might toggle when there is no data at the receiver input. `rx_is_lockedtoref` is a don't care when `rx_is_lockedtodata` is asserted. `rx_analogreset` must always be followed by `rx_digitalreset`.

Resetting the Receiver During Device Operation

Follow this reset sequence to reset the analog or digital blocks of the receiver at any point during the device operation. Use this reset to re-establish a link or after dynamic reconfiguration. The step numbers correspond to the numbers in the waveform.

1. Assert `rx_analogreset` and `rx_digitalreset`. Ensure that `rx_cal_busy` is low. You must reset the PCS by asserting `rx_digitalreset` every time you assert `rx_analogreset`.
2. Deassert `rx_analogreset` after a minimum duration of 40 ns.
3. Ensure `rx_is_lockedtodata` remains asserted before deasserting `rx_digitalreset`.
4. Deassert `rx_digitalreset` after `rx_is_lockedtodata` is asserted for a minimum duration of t_{LTD}.

Figure 4-6: Receiver Reset Sequence During Device Operation



Note: `rx_is_lockedtodata` might toggle when there is no data at the receiver input. `rx_is_lockedtoref` is a don't care when `rx_is_lockedtodata` is asserted.

Resetting the Transceiver in CDR Manual Lock Mode

You can use the clock data recovery (CDR) manual lock mode to override the default CDR automatic lock mode depending on your design requirements. The two control signals to enable and control the CDR in manual lock mode are `rx_set_locktoref` and `rx_set_locktodata`. Follow this sequence to reset your transceiver when the CDR is in manual lock mode.

Control Settings for CDR Manual Lock Mode

Use the following control settings to set the CDR lock mode:

Table 4-2: Control Settings for the CDR in Manual Lock Mode

rx_set_locktoref	rx_set_locktodata	CDR Lock Mode
0	0	Automatic
1	0	Manual-RX CDR LTR
X	1	Manual-RX CDR LTD

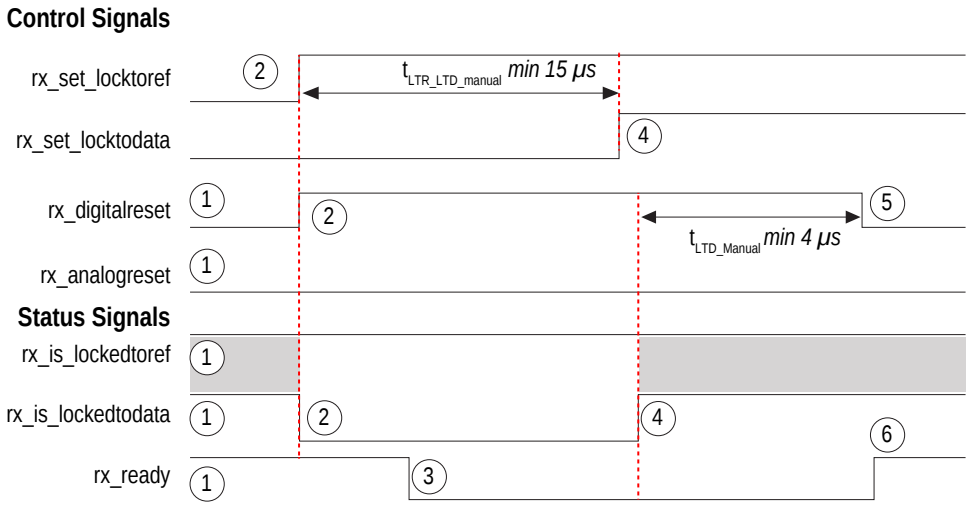
Resetting the Transceiver in CDR Manual Lock Mode

You can use the clock data recovery (CDR) manual lock mode to override the default CDR automatic lock mode depending on your design requirements. The two control signals to enable and control the CDR in manual lock mode are `rx_set_locktoref` and `rx_set_locktodata`.Follow this sequence to reset your transceiver when the CDR is in manual lock mode.

The numbers in the following figure correspond to the numbered list, which guides you through the steps to put the CDR in manual lock mode.

1. Make sure that the calibration is complete (`rx_cal_busy` is low) and the transceiver goes through the initial reset sequence. The `rx_digitalreset` and `rx_analogreset` signals should be low. The `rx_is_lockedtoref` is a don't care and can be either high or low. The `rx_is_lockedtodata` and `rx_ready` signals should be high, indicating that the transceiver is out of reset. Alternatively, you can start directly with the CDR in manual lock mode after the calibration is complete.
2. Assert the `rx_set_locktoref` signal high to switch the CDR to the lock-to-reference mode. The `rx_is_lockedtodata` status signal is deasserted. Assert the `rx_digitalreset` signal high at the same time or after `rx_set_lockedtoref` is asserted if you use the user-controlled reset. When the Transceiver PHY reset controller is used, the `rx_digitalreset` is automatically asserted.
3. After the `rx_digitalreset` signal gets asserted, the `rx_ready` status signal is deasserted.
4. Assert the `rx_set_locktodata` signal high after $t_{LTR\_LTD\_manual}$ to switch the CDR to the lock-to-data mode. The `rx_is_lockedtodata` status signal gets asserted when it acquires lock to the data. The `rx_is_lockedtoref` status signal can be a high or low and can be ignored.
5. Deassert the `rx_digitalreset` signal after t_{LTD_Manual}.
6. After the `rx_digitalreset` signal is deasserted, the `rx_ready` status signal gets asserted if you are using the Transceiver PHY Reset Controller, indicating that the receiver is now ready to receive data with the CDR in manual mode.

Figure 4-7: Reset Sequence Timing Diagram for Transceiver when CDR is in Manual Lock Mode



Transceiver Blocks Affected by Reset and Powerdown Signals

The following blocks are affected by the specified reset and powerdown signals. You must reset the digital PCS each time you reset the analog PMA or PLL. However, you can reset the digital PCS block alone.

Transceiver Block	pll_ powerdown	rx_digitalreset	rx_ analogreset	tx_digitalreset	tx_ analogreset
CMU PLL	Yes				
ATX PLL	Yes				
fPLL	Yes				
Receiver Standard PCS		Yes			
Receiver Enhanced PCS		Yes			
Receiver PMA			Yes		
Transmitter Standard PCS				Yes	
Transmitter Enhanced PCS				Yes	
Transmitter PMA					Yes
Receiver PCIe Gen3 PCS		Yes			
Transmitter PCIe Gen3 PCS				Yes	

Reset Signals for PLL, PMA, and PCS Blocks

The following table shows the signals the reset controller must assert to reset the specified blocks.

To Reset Blocks	Reset Controller Must Assert (In Order)
PLL or CDR	1. `pll_powerdown` 2. `tx_analogreset` 3. `tx_digitalreset`
TX PMA or RX PMA	1. `tx_analogreset` 2. `tx_digitalreset`
TX PCS or RX PCS	1. `tx_digitalreset`

Using the Altera Transceiver PHY Reset Controller

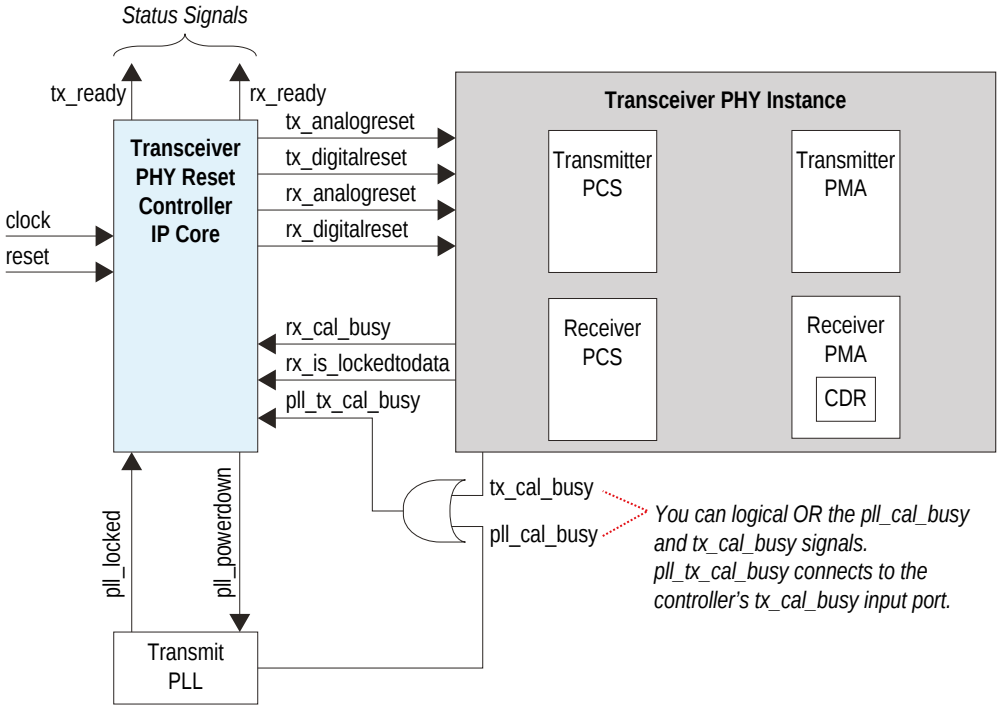
Altera's Transceiver PHY Reset Controller is a configurable IP core that resets transceivers mainly in response to PLL lock activity. You can use this IP core rather than creating your own user-coded reset controller. You can define a custom reset sequence for the IP core. You can also modify the IP cores's generated clear text Verilog HDL file to implement custom reset logic.

The Transceiver PHY Reset Controller handles all transceiver reset sequencing and supports the following options:

- Separate or shared reset controls per channel in response to PLL lock activity
- Separate controls for the TX and RX channels and PLLs
- Synchronization of the reset inputs
- Hysteresis for PLL locked status inputs
- Configurable reset timing
- Automatic or manual reset recovery mode in response to loss of PLL lock

You should create your own reset controller if the Transceiver PHY Reset Controller IP does not meet your requirements, especially when you require independent transceiver channel reset. The following figure illustrates the typical use of the Transceiver PHY Reset Controller in a design that includes a transceiver PHY instance and the transmit PLL.

Figure 4-8: Altera Transceiver PHY Reset Controller System Diagram



The Transceiver PHY Reset Controller IP core connects to the Transceiver PHY and the Transmit PLL. The Transceiver PHY Reset Controller IP core receives status from the Transceiver PHY and the Transmit PLL. Based on the status signals or the reset input, it generates TX and RX reset signals to the Transceiver PHY and TX PLL.

The`tx_ready signal` indicates whether the TX PMA exits the reset state, and if the TX PCS is ready to transmit data. The `rx_ready` signal indicates whether the RX PMA exits the reset state, and if the RX PCS is ready to receive data. You must monitor these signals to determine when the transmitter and receiver are out of the reset sequence.

Parameterizing the Transceiver PHY Reset Controller IP

This section lists steps to configure the Transceiver PHY Reset Controller IP Core in the MegaWizard Plug-In Manager. You can customize the following Transceiver PHY Reset Controller parameters for different modes of operation by clicking **Tools > MegaWizard Plug-In Manager**.

To parameterize and instantiate the Transceiver PHY Reset Controller IP core:

1. For **Which device family will you be using?**, select your target device from the list.
2. Click **Installed Plug-Ins > Interfaces > Transceiver PHY > Transceiver PHY Reset Controller**<*version*>.
3. Select the options required for your design. For a description of these options, refer to the **Transceiver PHY Reset Controller Parameters**.
4. Click **Finish**. The wizard generates files representing your parameterized IP variation for synthesis and simulation.

Transceiver PHY Reset Controller Parameters

The Quartus II software provides a GUI to define and instantiate a Transceiver PHY Reset Controller to reset transceiver PHY and external PLL.

Table 4-3: General Options

Name	Range	Description
Number of transceiver channels	`1-1000`	Specifies the number of channels that connect to the Transceiver PHY Reset Controller IP core. The upper limit of the range is determined by your FPGA architecture.
Number of TX PLLS	`1-1000`	Specifies the number of TX PLLs that connect to the Transceiver PHY Reset Controller IP core.
Input clock frequency	`1-500 MHz`	Input clock to the transceiver PHY Reset Controller IP core. The frequency of the input clock in MHz. The upper limit on the input clock frequency is the frequency achieved in timing closure.
Synchronize reset input	**On /Off**	When **On**, the Transceiver PHY Reset Controller synchronizes the reset to the Transceiver PHY Reset Controller input clock before driving it to the internal reset logic. When **Off**, the reset input is not synchronized.
Use fast reset for simulation	**On /Off**	When **On**, the Transceiver PHY Reset Controller uses reduced reset counters for simulation.
TX PLL		
Enable TX PLL channel reset control	**On /Off**	When **On**, the Transceiver PHY Reset Controller IP core enables the reset control of the TX PLL. When **Off**, the TX PLL reset control is disabled.
pll_powerdown duration	`1-999999999`	Specifies the duration of the PLL powerdown period in ns. The value is rounded up to the nearest clock cycle. The default value is 1000 ns.

Name	Range	Description
Synchronize reset input for PLL powerdown	**On /Off**	When **On**, the Transceiver PHY Reset Controller synchronizes the PLL powerdown reset with the Transceiver PHY Reset Controller input clock. When **Off**, the PLL powerdown reset is not synchronized.
TX Channel		
Enable TX channel reset control	**On /Off**	When **On**, the Transceiver PHY Reset Controller enables the control logic and associated status signals for TX reset. When **Off**, disables TX reset control and status signals.
Use separate TX reset per channel	**On /Off**	When **On**, each TX channel has a separate reset. When **Off**, the Transceiver PHY Reset Controller uses a shared TX reset controller for all channels.
TX digital reset mode	**Auto**, **Manual**, **Expose Port**	Specifies the Transceiver PHY Reset Controller behavior when the `pll_locked` signal is deasserted. The following modes are available: • **Auto :** The associated `tx_digital_reset` controller automatically resets whenever the `pll_locked` signal is deasserted. • **Manual :** The associated `tx_digital_reset` controller is not reset when the `pll_locked` signal is deasserted, allowing you to choose corrective action. • **Expose Port :** The `tx_manual` signal is a top-level signal of the IP Core. You can dynamically change this port to AUTO or Manual. (1= Manual , 0 = AUTO)
tx_digitalreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `tx_digitalreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. The default value is 20 ns.
pll_locked input hysteresis	`1-999999999`	Specifies the amount of hysteresis in ns to add to the `pll_locked` status input to filter spurious unreliable assertions of the `pll_locked` signal. A value of 0 adds no hysteresis. A higher value filters glitches on the `pll_locked` signal.
RX Channel		
Enable RX channel reset control	**On /Off**	When **On**, the Transceiver PHY Reset Controller enables the control logic and associated status signals for RX reset. When **Off**, disables RX reset control and status signals.
Use separate RX reset per channel	**On /Off**	When **On**, each RX channel has a separate reset input. When **Off**, uses a shared RX reset controller for all channels.

Name	Range	Description
RX digital reset mode	**Auto**, **Manual**, **Expose Port**	Specifies the Transceiver PHY Reset Controller behavior when the PLL lock signal is deasserted. The following modes are available: • **Auto :** The associated `rx_digital_reset` controller automatically resets whenever the `rx_is_lockedtodata` signal is deasserted. • **Manual :** The associated `rx_digital_reset` controller is not reset when the `rx_is_lockedtodata` signal is deasserted, allowing you to choose corrective action. • **Expose Port :** The `rx_manual` signal is a top-level signal of the IP Core. If the core includes separate reset control for each RX channel, each RX channel uses its respective `rx_is_lockedtodata` signal for automatic reset control; otherwise, the inputs are ANDed to provide internal status for the shared reset controller.
rx_analogreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `rx_analogreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. The default value is 40 ns.
rx_digitalreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `rx_digitalreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. The default value is 4000 ns.

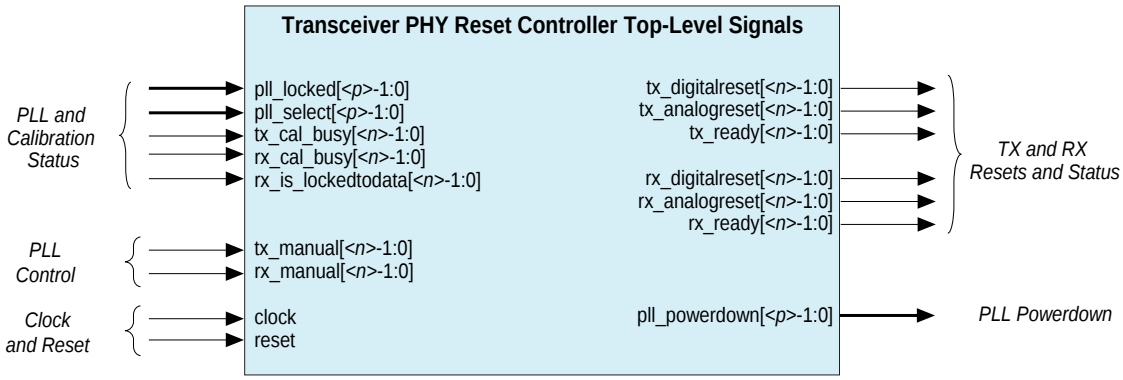
Transceiver PHY Reset Controller Interfaces

This section describes the top-level signals for the Transceiver PHY Reset Controller IP core.

The following figure illustrates the top-level signals of the Transceiver PHY Reset Controller IP core. Many of the signals in the figure become buses if you choose separate reset controls. The variables in the figure represent the following parameters:

- <*n*>—The number of lanes
- <*p*>—The number of PLLs

Figure 4-9: Transceiver PHY Reset Controller IP Core Top-Level Signals



Note: PLL control is available when you enable the **Expose Port** parameter.

Table 4-4: Top-Level Signals

This table describes the signals in the above figure in the order that they are shown in the figure.

Signal Name	Direction	Clock Domain	Description
`pll_locked[<p>-1:0]`	Input	Asynchronous status from TX PLL.	Provides the PLL locked status input from each PLL. When asserted, indicates that the TX PLL is locked. When deasserted, the PLL is not locked. There is one signal per PLL.
`pll_select[<p>-1:0]`	Input	Synchronous to the Transceiver PHY Reset Controller input clock. Set to zero when not using multiple PLLs.	When you select **Use separate TX reset per channel**, this bus provides enough inputs to specify an index for each `pll_locked` signal to listen to for each channel. When **Use separate TX reset per channel** is disabled, the `pll_select` signal is used for all channels.
`tx_cal_busy[<n>-1:0]`	Input	N/A	This is the calibration status signal that results from the logical OR of `pll_cal_busy` and `tx_cal_busy` signals. The signal goes high when either the TX PLL or Transceiver PHY calibration is active. The signal goes low when calibration is completed. This signal gates the TX reset sequence. The width of this signals depends on the number of TX channels.
`rx_cal_busy[<n>-1:0]`	Input	Asynchronous	This is calibration status signal from the Transceiver PHY IP core. When asserted, calibration is active. When deasserted, calibration has completed. This signal gates the RX reset sequence. The width of this signals depends on the number of RX channels.

Signal Name	Direction	Clock Domain	Description
rx_is_lockedtodata[<n>-1:0]	Input	CDR	Provides the rx_is_lockedtodata status from each RX CDR. When asserted, indicates that a particular RX CDR is ready to receive input data. If you do not choose separate controls for the RX channels, these inputs are ANDed together internally to provide a single status signal.
tx_manual[<n>-1:0]	Input	Asynchronous	This optional signal places tx_digitalreset controller under automatic or manual control. When asserted, the associated tx_digitalreset controller logic does not automatically respond to deassertion of the pll_locked signal. However, the initial tx_digitalreset sequence still requires a one-time rising edge on pll_locked before proceeding. When deasserted, the associated tx_digitalreset controller automatically begins its reset sequence whenever the selected pll_locked signal is deasserted.
rx_manual[<n> -1:0]	Input	Asynchronous	This optional signal places rx_digitalreset logic controller under automatic or manual control. In manual mode, the rx_digitalreset controller does not respond to the assertion or deassertion of the rx_is_lockedtodata signal. The rx_digitalreset controller asserts rx_ready when the pll_locked signal is asserted.
clock	Input	N/A	A free running system clock input to the Transceiver PHY Reset Controller from which all internal logic is driven. If a free running clock is not available, hold resets until the system clock is stable.
reset	Input	Asynchronous	Asynchronous reset input to the Transceiver PHY Reset Controller. When asserted, all configured reset outputs are asserted. Holding the reset input signal asserted holds all other reset outputs asserted.

Signal Name	Direction	Clock Domain	Description
tx_digital-reset[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Digital reset for TX. The width of this signal depends on the number of TX channels. This signal is asserted when any of the following conditions is true: • reset is asserted • pll_powerdown is asserted • pll_cal_busy is asserted • tx_cal_busy is asserted • PLL has not reached the initial lock (pll_locked deasserted) • pll_locked is deasserted and tx_manual is deasserted When all of these conditions are false, the reset counter begins its countdown for deassertion of tx_digital_reset If your design includes bonded TX PCS channels, refer to Timing Constraints for Reset Signals when Using Bonded PCS Channels for a SDC constraint you must include in your design.
tx_analogreset[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Analog reset for TX channels. This signal follows pll_powerdown and is deasserted after the TX PLL comes out the reset and locks to the input reference clock.
tx_ready[<n> 1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Status signal to indicate when the TX reset sequence is complete. This signal is deasserted while the TX reset is active. It is asserted a few clock cycles after the deassertion of tx_digitalreset. Some protocol implementations may require you to monitor this signal prior to sending data. The width of this signal depends on the number of TX channels.
rx_digital-reset[<n> - 1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Digital reset for RX. The width of this signal depends on the number of channels. This signal is asserted when any of the following conditions is true: • reset is asserted • rx_analogreset is asserted • rx_cal_busy is asserted • rx_is_lockedtodata is deasserted and rx_manual is deasserted When all of these conditions are false, the reset counter begins its countdown for deassertion of rx_digitalreset.

Signal Name	Direction	Clock Domain	Description
rx_ready[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Status signal to indicate when the RX reset sequence is complete. This signal is deasserted while the RX reset is active. It is asserted a few clock cycles after the deassertion of rx_digitalreset. Some protocol implementations may require you to monitor this signal prior to sending data. The width of this signal depends on the number of channels.
pll_powerdown[<p>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Asserted to power down a transceiver PLL circuit. When asserted, the selected TX PLL is reset.

Transceiver PHY Reset Controller Resource Utilization

This section describes the estimated device resource utilization for two configurations of the transceiver PHY reset controller. The exact resource count varies by Quartus II version number, as well as by optimization options.

Table 4-5: Reset Controller Resource Utilization

Configuration	Combination ALUTs	Logic Registers
Single transceiver channel	approximately 50	approximately 50
4 Transceiver channels, shared TX reset, separate RX resets	approximately 100	approximately 150

Using a User-Coded Reset Controller

You can choose to design your own user-coded reset controller rather than using Altera's Transceiver PHY Reset Controller IP core. Your user coded reset controller must provide the following functionality for the recommended reset sequence:

- A clock signal input for your reset logic
- Holds the transceiver channels in reset by asserting the appropriate reset control signals
- Checks the PLL status (for example, checks the status of pll_locked and pll_cal_busy)

Note: You must ensure a stable reference clock is present at the PLL transmitter before releasing PLL powerdown (pll_powerdown).

User-Coded Reset Controller Signals

Refer to the signals in the following figure and table for implementation of a user-coded reset controller.

Figure 4-10: User-Coded Reset Controller, Transceiver PHY, and TX PLL Interaction

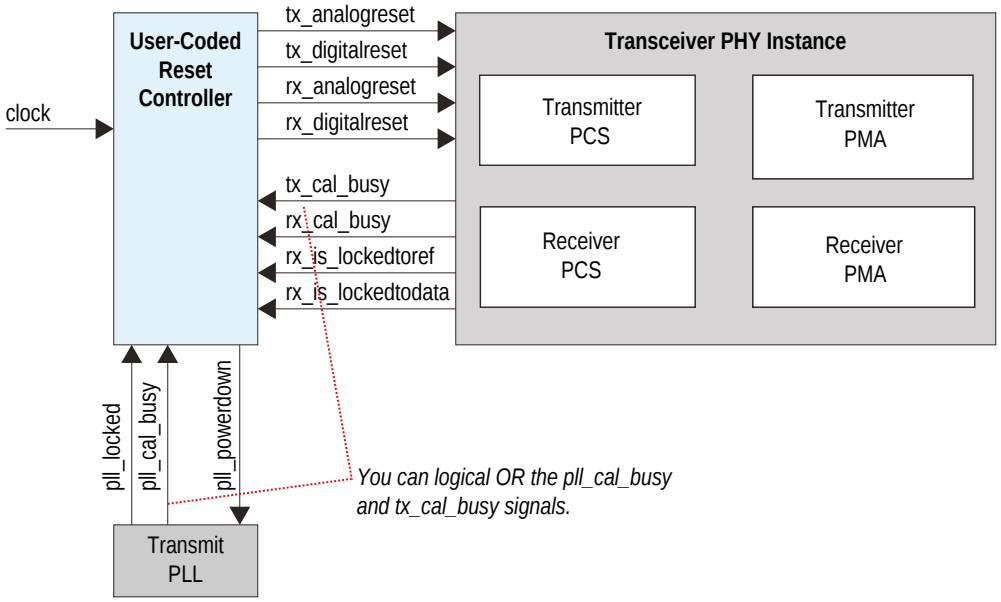


Table 4-6: User-coded Reset Controller, Transceiver PHY, and ATXPLL Signals

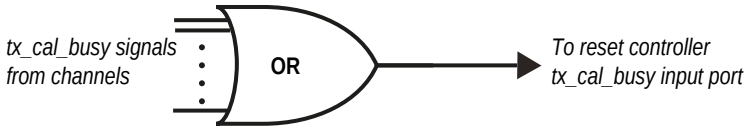
Signal Name	Direction	Description
pll_powerdown	Output	Resets the TX PLL when asserted high
tx_analogreset	Output	Resets the TX PMA when asserted high
tx_digitalreset	Output	Resets the TX PCS when asserted high
rx_analogreset	Output	Resets the RX PMA when asserted high
rx_digitalreset	Output	Resets the RX PCS when asserted high
clock	Input	Clock signal for the user-coded reset controller. You can use the system clock without synchronizing it to the PHY parallel clock. The upper limit on the input clock frequency is the frequency achieved in timing closure.
pll_cal_busy	Input	A high on this signal indicates the PLL is being calibrated.
pll_locked	Input	A high on this signal indicates that the TX PLL is locked to the ref clock
tx_cal_busy	Input	A high on this signal indicates that TX calibration is active. If you have multiple PLLs, you can OR their pll_cal_busy signals together.
rx_is_lockedto-data	Input	A high on this signal indicates that the RX CDR is in the lock-to-data (LTD) mode

Signal Name	Direction	Description
rx_cal_busy	Input	A high on this signal indicates that RX calibration is active
rx_is_lockedtoref	Input	A high on this signal indicates that the RX CDR is in the lock-to-reference (LTR) mode. This signal may toggle or be deasserted when the CDR is in LTD mode.

Combining Status or PLL Lock Signals

You can combine multiple PHY status signals before feeding into the reset controller as shown below.

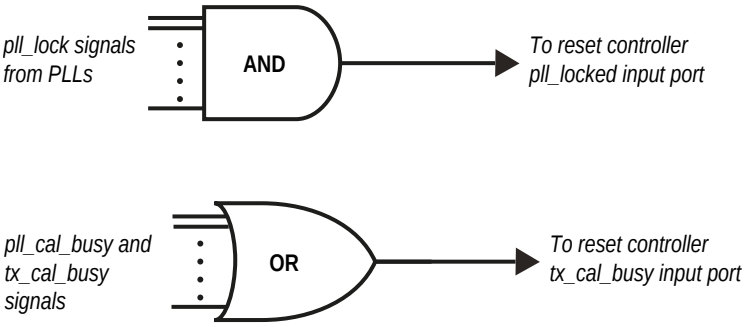
Figure 4-11: Combining Multiple PHY Status Signals



Note: This configuration also applies to the rx_cal_busy signals.

When using multiple PLLs, you can logical AND the pll_locked signals feeding the reset controller. Similarly, you can logical OR the pll_cal_busy signals to the reset controller tx_cal_busy port as shown below.

Figure 4-12: Multiple PLL Configuration

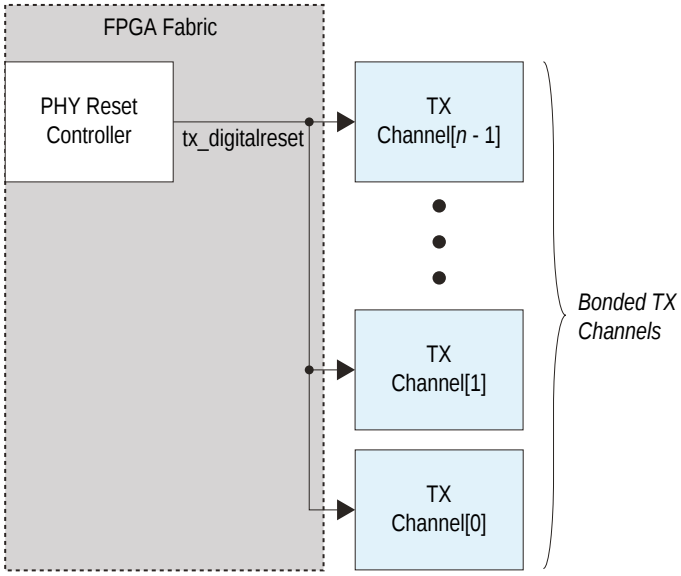


Resetting different channels separately requires multiple reset controllers. For example, a group of channels configured for Interlaken requires a separate reset controller from another group of channels that are configured for optical communication.

Timing Constraints for Bonded PCS and PMA Channels

For designs that use bonded TX PCS and PMA channels, the digital reset signal to all TX channels within a bonded group must meet a maximum skew tolerance imposed by physical routing. This skew tolerance is one-half the TX parallel clock cycle (tx_clockout). This requirement is not necessary for bonded TX PMA-only channel or for RX PCS channels.

Figure 4-13: Physical Routing Delay Skew in Bonded Channels



You must provide a Synopsys Design Constraint (SDC) for the reset signals to guarantee that your design meets timing. The Quartus II software generates an .sdc file when you generate the Transceiver Native PHY IP. This .sdc contains constraints for setting the maximum skew and false paths. This constraint accounts for the skew introduced by routing delays inherent within the FPGA. This skew is present whether you tie all `tx_digitalresets` together, or you control them separately. If your design includes the Transceiver PHY Reset Controller IP core, you can substitute your instance and interface names for the generic names shown in the example.

Example 4-1: SDC Constraint for TX Digital Reset When Bonded Clocks Are Used

```
set_max_skew -from *<IP_INSTANCE_NAME> *tx_digitalreset*r_reset
-to *pld_pcs_interface* <1/2 coreclk period in ps>
```

In the above example, you must make the following substitutions:

- *<IP_INSTANCE_NAME>*—substitute the name of your reset controller IP instance or PHY IP instance
- *<½ coreclk period in ps>*—substitute the ½ the clock period of your design in picoseconds

If your design has custom reset logic, replace the `*<IP_INSTANCE_NAME>*tx_digital-reset*r_reset` with the source register for the TX PCS reset signal, `tx_digitalreset`.

For more information about the `set_max_skew` constraint, refer to the *SDC and TimeQuest API Reference Manual*.

Related Information

SDC and TimeQuest API Reference Manual

Arria 10 Transceiver PHY Architecture

5

2013.12.02

UG-A10XCVR

Subscribe Send Feedback

Arria 10 PMA Architecture

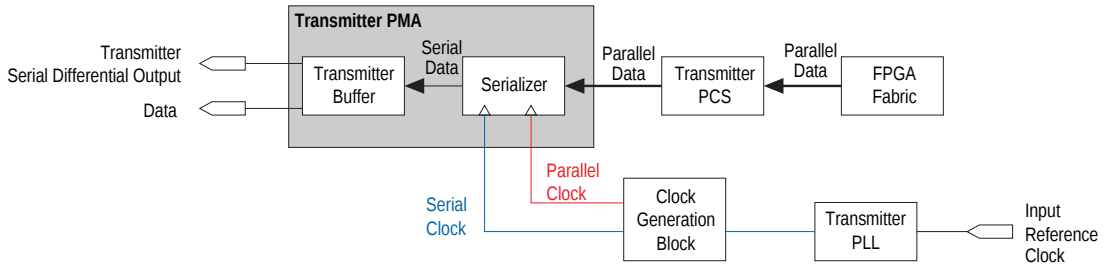
The Physical Medium Attachment (PMA) acts as the analog front end for the Arria 10 transceivers.

The PMA receives and transmits high-speed serial data depending on the transceiver channel configuration. All serial data transmitted and received passes through the PMA.

Transmitter

The transmitter takes the parallel data and serializes it to create a high-speed serial data stream. The transmitter portion of the PMA is composed of the transmitter serializer and the transmitter buffer. The serializer clock is provided from the transmitter PLL.

Figure 5-1: Transmitter PMA Block Diagram



Serializer

The serializer converts the incoming low-speed parallel data from the transceiver PCS or FPGA fabric to high-speed serial data and sends the data to the transmitter buffer.

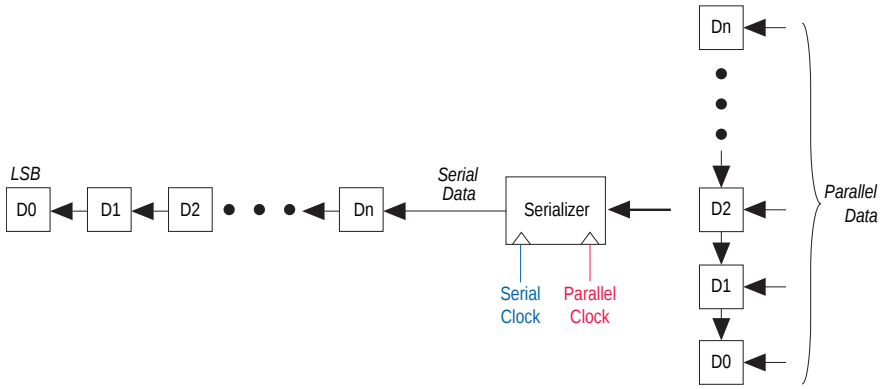
The channel serializer supports the following serialization factors: 8, 10, 16, 20, 32, 40 and 64.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 5-2: Serializer Block

The serializer block sends out the least significant bit (LSB) of the input data first.

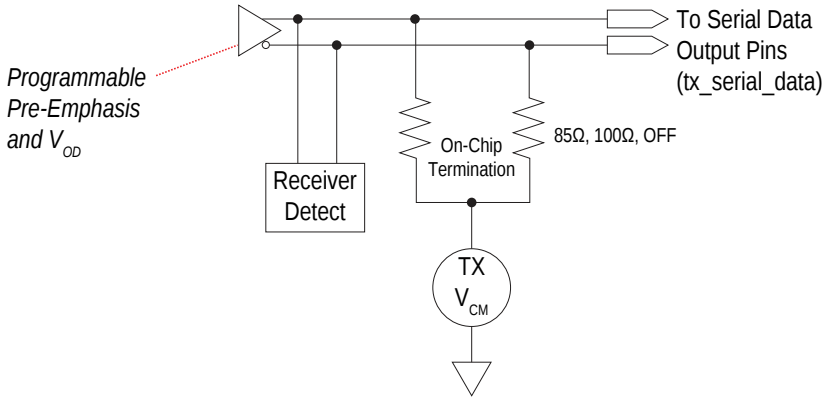


Transmitter Buffer

The transmitter buffer includes the following circuitry to improve signal integrity:

- programmable differential output voltage (V_{OD})
  - main tap
- programmable four-tap pre-emphasis circuitry
  - two pre-cursor taps
  - two post-cursor taps
- internal termination circuitry
- receiver detect capability to support PCI Express and Quick Path Interconnect (QPI) configurations

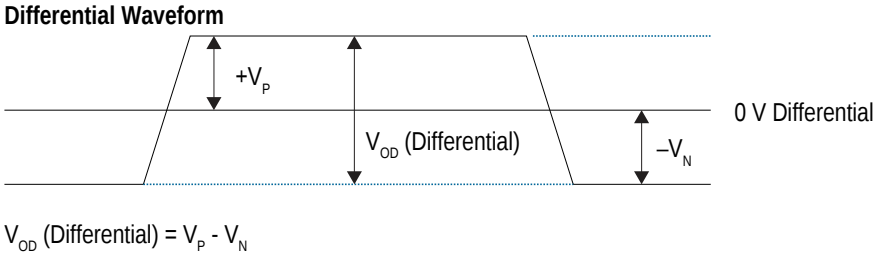
Figure 5-3: Transmitter Buffer



Programmable Output Differential Voltage

You can program the differential output voltage to handle different channel losses and receiver requirements. There are 32 differential V_{OD} settings up to 1000 mV. The step size is 31.25 mV.

Figure 5-4: V_{OD} (Differential) Signal Level



Programmable Pre-Emphasis

Pre-emphasis can maximize the eye at the far-end receiver. The programmable pre-emphasis module in each transmit buffer amplifies high frequencies in the transmit data signal, to compensate for attenuation in the transmission media.

The pre-tap de-emphasizes the bit before the transition and emphasizes the remaining bits. A different polarity on pre-tap does the opposite.

Figure 5-5: Effect of the 1st Post-Tap

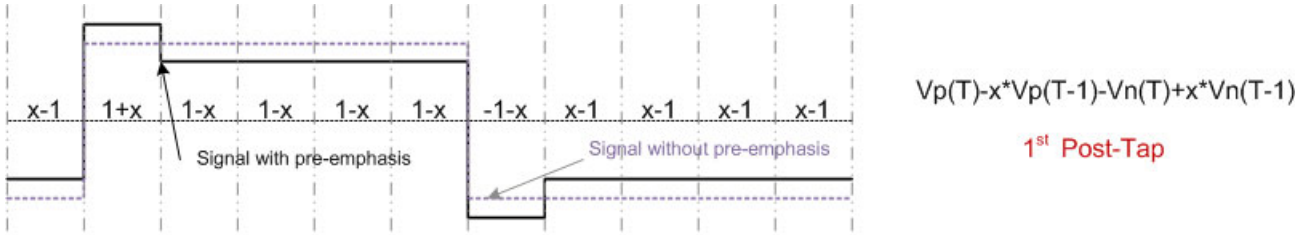


Figure 5-6: Effect of the Pre-Tap

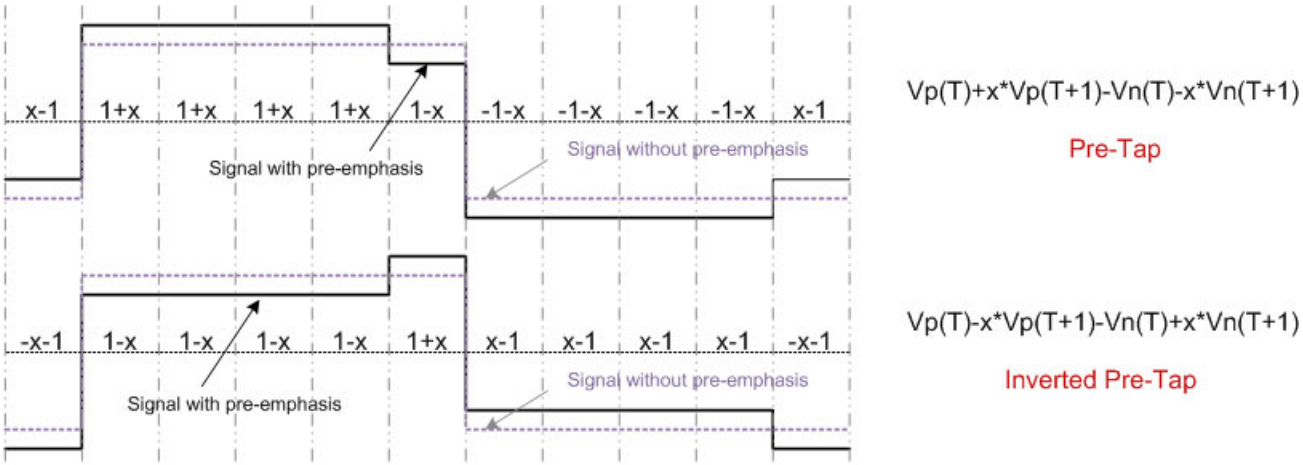


Table 5-1: Pre-Emphasis Taps

All four pre-emphasis taps provide inversion control, shown by negative values.

Pre-Emphasis Tap	Number of Settings	Channel Loss Compensation (dB)
Second pre-tap	15	2.31
First pre-tap	33	6.62
First post-tap	51	15.56
Second post-tap	25	4.44

You can set pre-emphasis taps through the Avalon-MM registers, which will be available in a future release of the Quartus II software.

Related Information

For more information about pre-emphasis, refer to AN-602.

Programmable Transmitter On-Chip Termination (OCT)

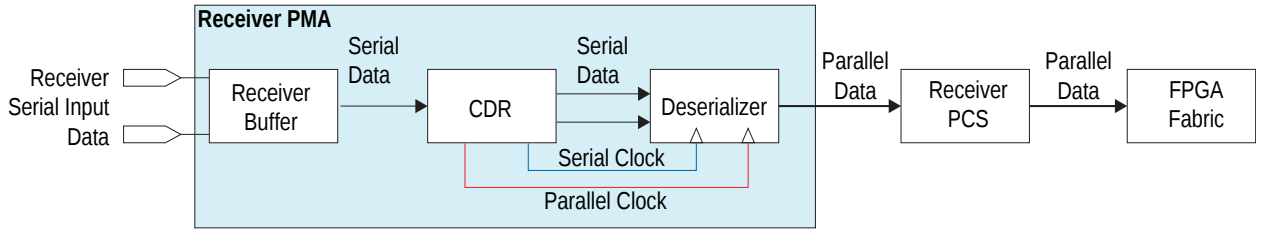
Transmitter buffers include programmable on-chip differential termination of 85Ω, 100Ω, or OFF.

Receiver

The receiver deserializes the high-speed serial data, creates a parallel data stream for either the receiver PCS or the FPGA fabric, and recovers the clock information from the received data.

The receiver portion of the PMA is comprised of the receiver buffer, the clock data recovery (CDR) unit, and the deserializer.

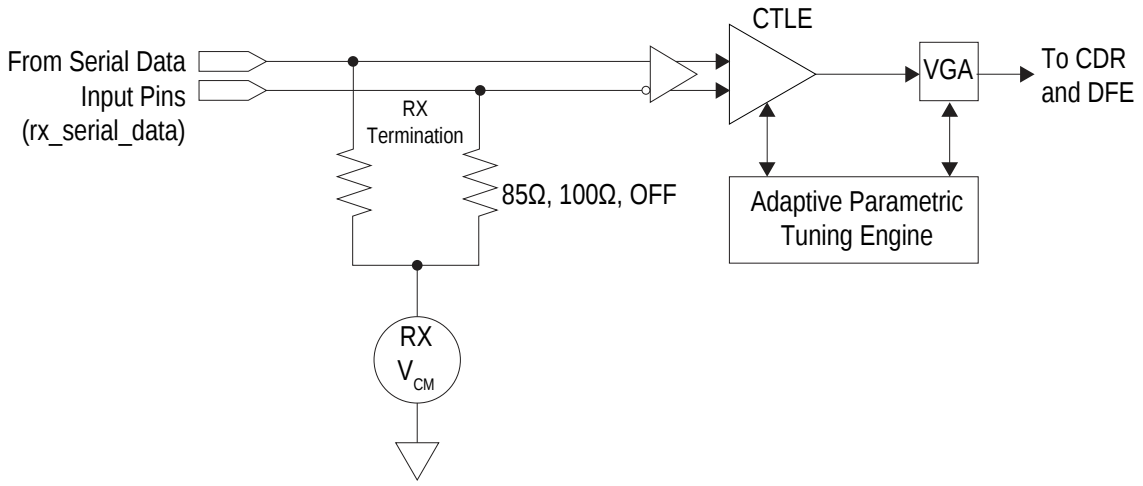
Figure 5-7: Receiver PMA Block Diagram



Receiver Buffer

The receiver input buffer receives serial data from `rx_serial_data` and feeds the serial data to the clock data recovery (CDR) unit and deserializer.

Figure 5-8: Receiver Buffer



The receiver buffer supports the following features:

- Programmable common mode voltage (V_{CM})
- Programmable differential On-Chip Termination (OCT)
- Signal Detector
- Continuous Time Linear Equalization (CTLE)
- Variable Gain Amplifiers (VGA)
- Adaptive Parametric Tuning Engine
- Decision Feedback Equalization (DFE)

Programmable Common Mode Voltage (V_{CM})

The receiver buffer has on-chip biasing circuitry to establish the required V_{CM} at the receiver input.

The Quartus II software automatically chooses the optimal setting for RX V_{CM}.

Note: On-chip biasing circuitry is available only if you select OCT. If you select external termination, you must implement off-chip biasing circuitry to establish the V_{CM} at the receiver input buffer.

Programmable Differential On-Chip Termination (OCT)

Receiver buffers include programmable on-chip differential termination of 85Ω, 100Ω, or OFF.

You can disable OCT and use external termination. If you select external termination, the receiver common mode is tri-stated. Common mode is based on the external termination connection. You will also need to implement off-chip biasing circuitry to establish the V_{CM} at the receiver buffer.

Signal Detector

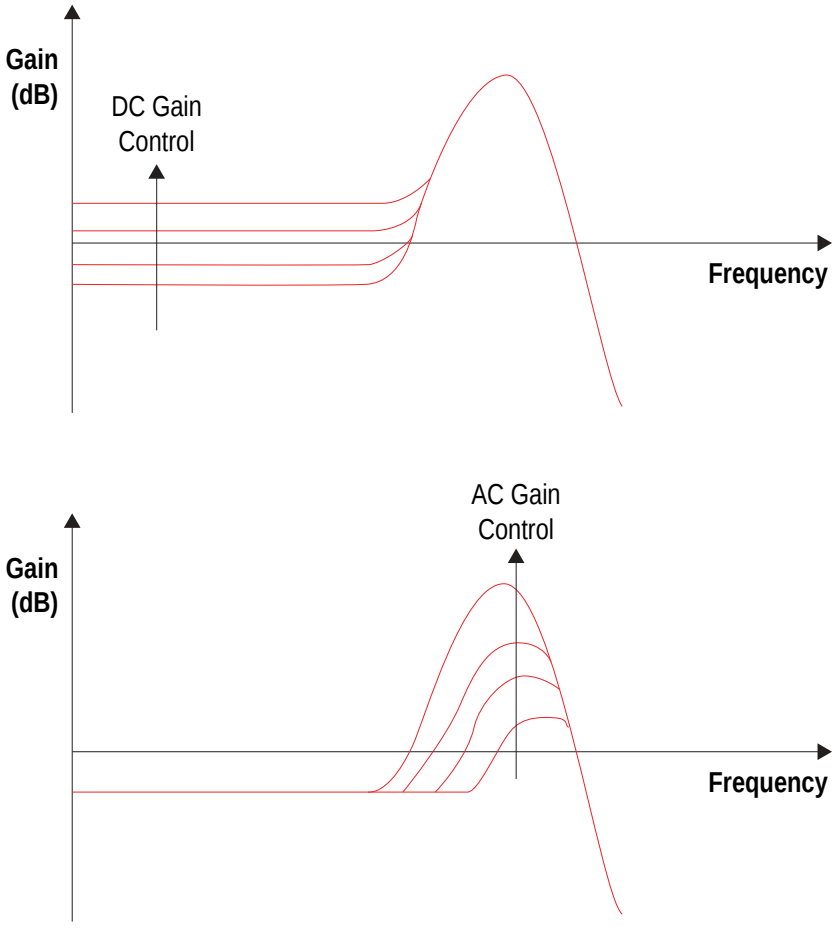
You can enable the optional signal threshold detection circuitry. If enabled, this option senses whether the signal level present at the receiver input buffer is above the signal detect threshold voltage that you specified in the assignment editor in Quartus II.

Continuous Time Linear Equalization (CTLE)

Each receiver buffer has independently programmable equalization circuits that amplify the high-frequency component of the incoming signal, compensating for the low-pass characteristics of the physical medium. The CTLE supports both DC and AC gain.

DC gain circuitry provides an equal amplification to the incoming signal across the frequency spectrum. AC gain circuitry provides amplification to the high-frequency spectrum gain of the incoming signal.

Figure 5-9: CTLE AC and DC Gain Conceptualization



Note: Final equalization curves will be available in the Arria 10 datasheet.

Arria 10 transceiver channels have dual mode CTLE-high gain mode and high data rate mode. In high gain mode, CTLE provides up to 16 dB programmable AC gain for data rates up to 12.5 Gbps. In high data rate mode, CTLE provides up to 12 dB programmable AC gain for data rates up to 28.1 Gbps.

CTLE in Arria 10 transceivers also provides different bandwidth settings for different data rates. High gain mode has the following options:

- low (up to 6.25 Gbps)
- high (up to 12.5 Gbps)

High data rate mode provides gain for higher data rate systems up to 28.1 Gbps. You can choose from a range of programmable peaking frequencies for this mode of the CTLE.

The CTLE operates in both fixed and adaptive equalization modes. Adaptive equalization is controlled by the Adaptive Parametric Tuning Engine. You can select the CTLE mode of operation through the Avalon Memory-Mapped (Avalon-MM) registers.

Related Information

- **Arria 10 Register Map**
- **Arria 10 Device Datasheet**

Variable Gain Amplifier (VGA)

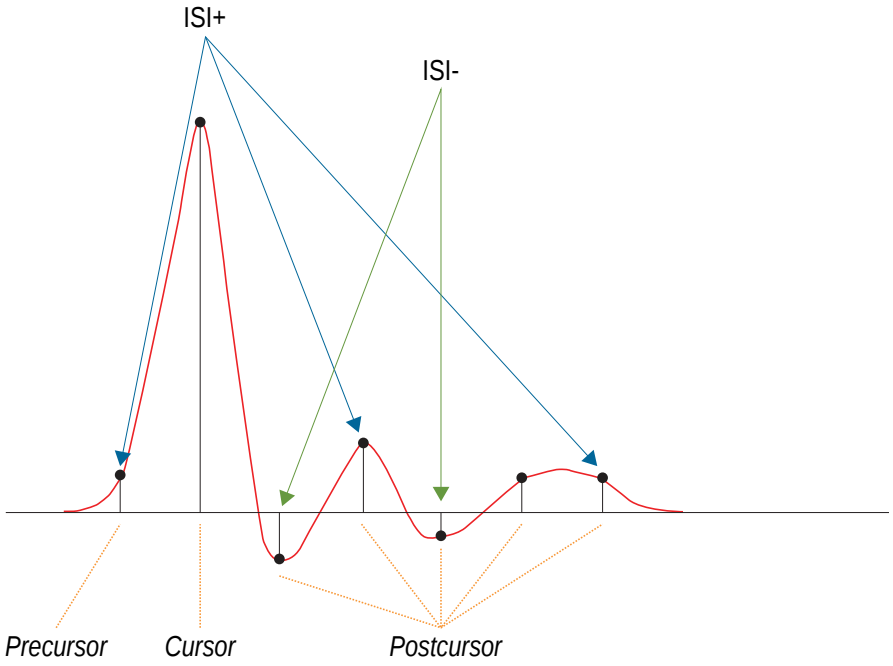
Arria 10 channels have a variable gain amplifier to optimize the signal amplitude prior to the CDR sampling. The VGA operates both in fixed and adaptive modes. Adaptive VGA is controlled by the Adaptive Parametric Tuning Engine.

Decision Feedback Equalization (DFE)

DFE amplifies the high frequency components of a signal without amplifying the noise content. It compensates for inter-symbol interference (ISI). DFE minimizes post-cursor ISI by adding or subtracting weighted versions of the previously received bits, from the current bit. DFE can work in synchronization with the TX pre-emphasis and downstream RX CTLE, to enable the RX CDR to receive the correct data that was transmitted through a lossy and noisy backplane.

An advantage of DFE over CTLE is improved Signal to Noise Ratio (SNR). That is, DFE amplifies the power of the high frequency components without amplifying the noise power.

Figure 5-10: Signal ISI



Notes:
- An ideal pulse response is a single data point at the cursor.
- Real world pulse response is non-zero before the cursor (precursor) and after the cursor (postcursor).
- ISI occurs when the data sampled at precursor or postcursor is not zero.

The DFE circuit stores delayed versions of the data. The stored bit is multiplied by a coefficient and then summed with the incoming signal. The polarity of each coefficient is programmable.

The DFE architecture supports seven fixed taps and four floating taps.

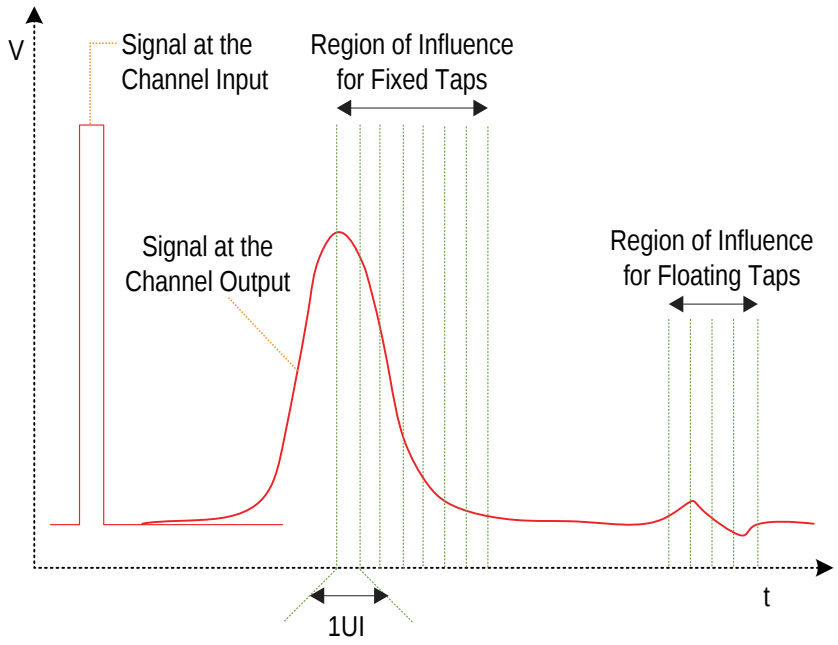
The modes of operation for DFE are:

- Continuous adaptation - Continuously runs DFE adaptation
- Triggered adaptation - Stops adaptation once DFE adaptation is complete
- Manual - You can manually set the fixed tap and floating tap values through the Avalon-MM registers

The seven fixed taps translate to the DFE capable of removing the ISI from the next 7 bits, beginning from the current bit.

The pulse at the output of the channel shows a long decaying tail because of the frequency-dependent losses, along with other signal quality degradation.

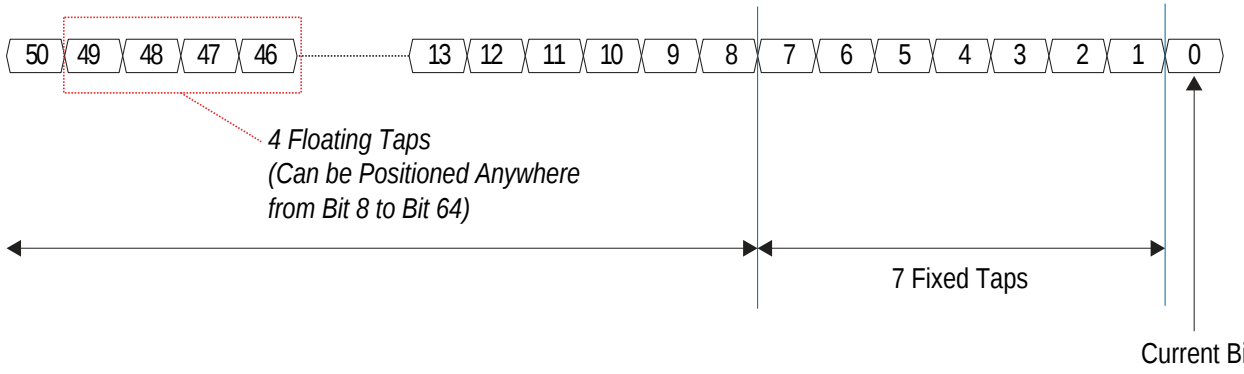
Figure 5-11: Channel Pulse Response



Reflections also appear on the post cursor tail at a later time. These reflections occur due to impedance discontinuities at various points along the signal path. Some such points are between the package and daughter card, daughter card and connecter, connector and backplane, and so on. Increasing the number of fixed taps can help mitigate these reflections. However, instead of having a large number of fixed taps, it is more efficient to have a few fixed taps and some floating taps which can be positioned as needed.

DFE in Arria 10 transceivers has four floating taps, and their location can be selected anywhere from 8 UI up to 64 UI.

Figure 5-12: DFE Floating and Fixed Taps

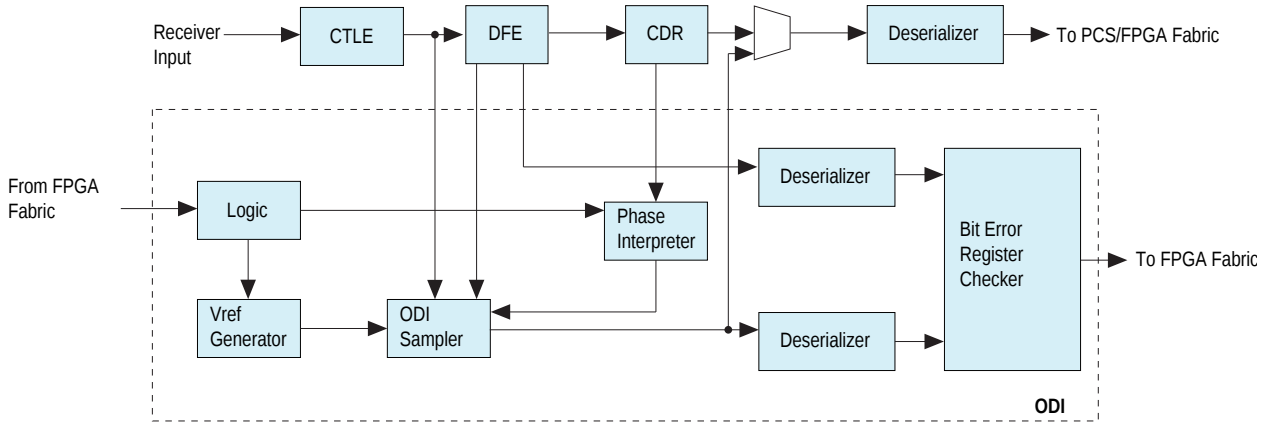


On-Die Instrumentation (ODI)

On-Die Instrumentation (ODI) provides on-chip eye monitoring capabilities (EyeQ).

This capability helps to both optimize link equalization parameters during board bring-up and provide in-system link diagnostics.

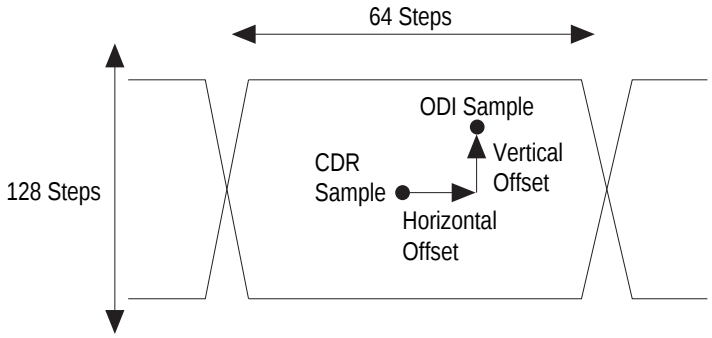
Figure 5-13: Receiver and ODI Architecture



ODI in Arria 10 transceivers uses a phase interpreter to generate a horizontal offset and the voltage reference (Vref) generator to generate a vertical offset to get the ODI sample. By comparing the bit difference between the CDR sample and the ODI sample in the bit error register checker, ODI can monitor link margin over live traffic.

ODI in Arria 10 transceivers provides 64 horizontal steps and 128 vertical steps to monitor eye margin. You can set both horizontal and vertical steps through the Avalon-MM registers.

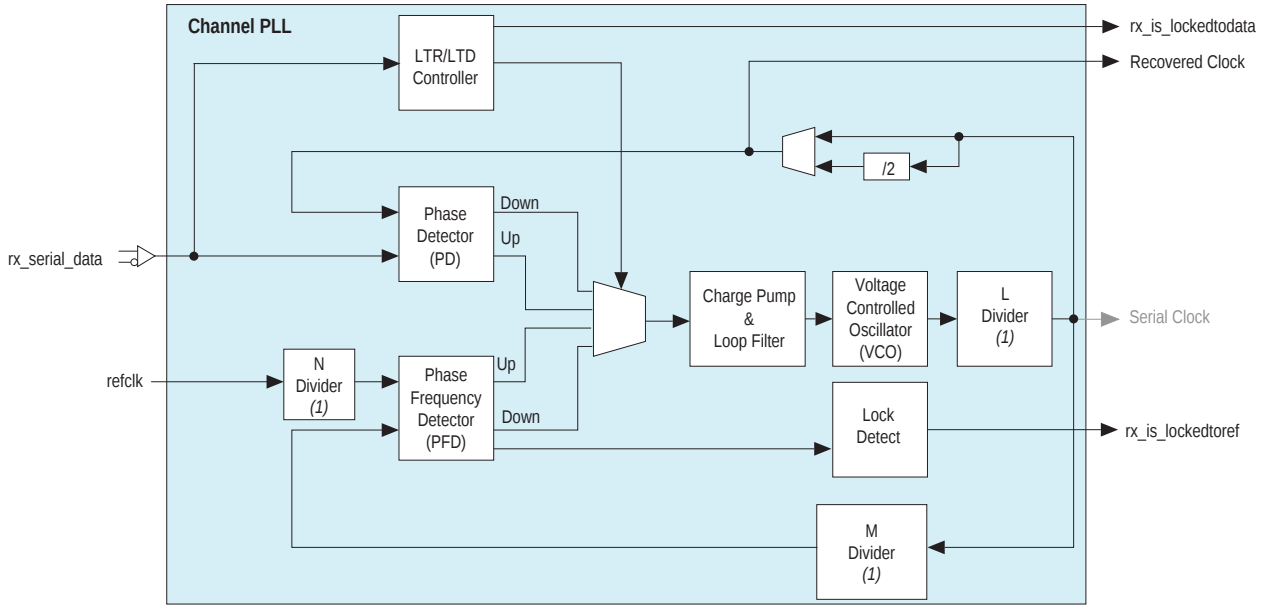
Figure 5-14: Bit Error Register Checker



Clock Data Recovery (CDR) Unit

The PMA of each channel includes a channel PLL that you can configure as a receiver clock data recovery (CDR) for the receiver, or a clock multiplier unit (CMU) PLL for the transmitter.

Figure 5-15: Channel PLL Configured as CDR



Note:
1. The Quartus II software automatically chooses the optimal values.

Lock-to-Reference Mode

In LTR mode, the phase frequency detector (PFD) in the CDR tracks the receiver input reference clock. The PFD controls the charge pump that tunes the VCO in the CDR. The `rx_is_lockedtoref` status signal is asserted active high to indicate that the CDR has locked to the phase and frequency of the receiver input reference clock.

Note: The phase detector (PD) is inactive in LTR mode.

Lock-to-Data Mode

During normal operation, the CDR must be in LTD mode to recover the clock from the incoming serial data. In LTD mode, the PD in the CDR tracks the incoming serial data at the receiver input. Depending on the phase difference between the incoming data and the CDR output clock, the PD controls the CDR charge pump that tunes the VCO.

Note: The PFD is inactive in LTD mode. The `rx_is_lockedtoref` status signal toggles randomly and is not significant in LTD mode.

After switching to LTD mode, the `rx_is_lockedtodata` status signal is asserted. The actual lock time depends on the transition density of the incoming data and the parts per million (ppm) difference between the receiver input reference clock and the upstream transmitter reference clock. The `rx_is_lockedtodata` signal toggles until the CDR sees valid data; therefore, you should hold receiver PCS logic in reset (`rx_digitalreset`) for a minimum of 4 µs after `rx_is_lockedtodata` remains continuously asserted.

CDR Lock Modes

You can configure the CDR in either automatic lock mode or manual lock mode. By default, the Quartus II software configures the CDR in automatic lock mode.

Automatic Lock Mode

In automatic lock mode, the CDR initially locks to the input reference clock (LTR mode). After the CDR locks to the input reference clock, the CDR locks to the incoming serial data (LTD mode) when the following conditions are met:

- The signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer when `rx_std_signaldetect` is enabled..
- The CDR output clock is within the configured ppm frequency threshold setting with respect to the input reference clock (frequency locked).
- The CDR output clock and the input reference clock are phase matched within approximately 0.08 unit interval (UI) (phase locked).

If the CDR does not stay locked to data because of frequency drift or severe amplitude attenuation, the CDR switches back to LTR mode.

Manual Lock Mode

The PPM detector and phase relationship detector reaction times can be too long for some applications that require faster CDR lock time. You can manually control the CDR to reduce its lock time using two optional input ports (`rx_set_locktoref` and `rx_set_locktodata`).

Table 5-2: Relationship Between Optional Input Ports and the CDR Lock Mode

rx_set_locktoref	rx_set_locktodata	CDR Lock Mode
0	0	Automatic
1	0	Manual-RX CDR LTR
X	1	Manual-RX CDR LTD

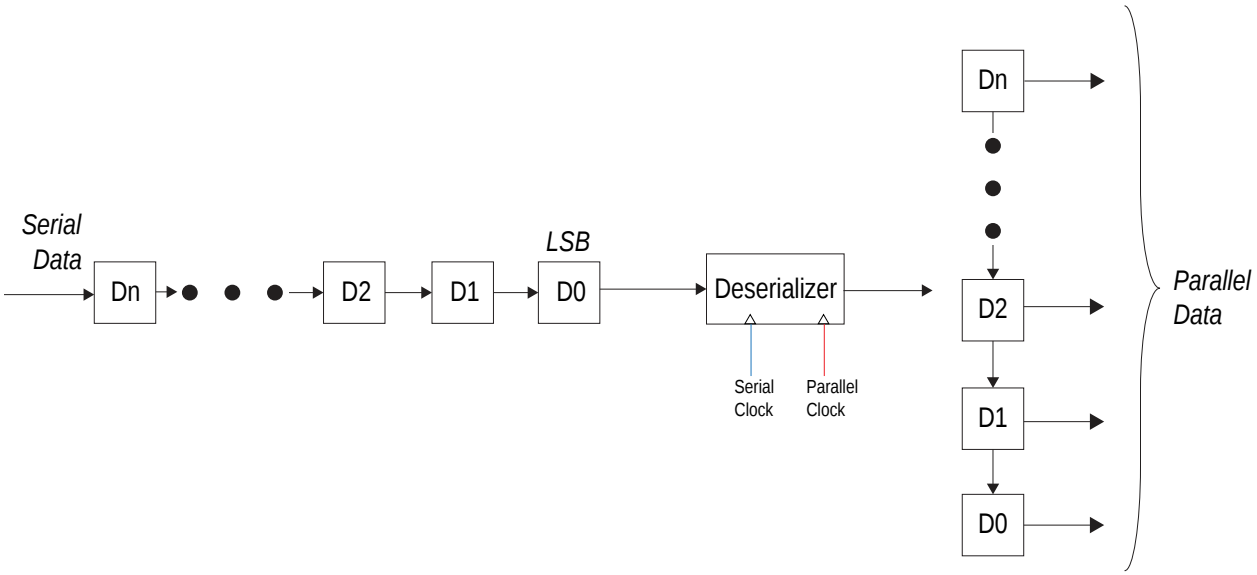
Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes the data using the low-speed parallel recovered clock. The deserializer forwards the deserialized data to the receiver PCS or FPGA fabric.

The deserializer supports the following deserialization factors: 8, 10, 16, 20, 32, 40, and 64.

Figure 5-16: Deserializer Block Diagram

The deserializer block sends out the LSB of the input data first.



Loopback

The Arria 10 PMA supports serial, diagnostic, and reverse loopback paths.

Figure 5-17: Serial Loopback Path

The serial loopback path sets the CDR to recover the data from serializer while data from receiver serial input pin is ignored by CDR. The transmitter buffer sends data normally.

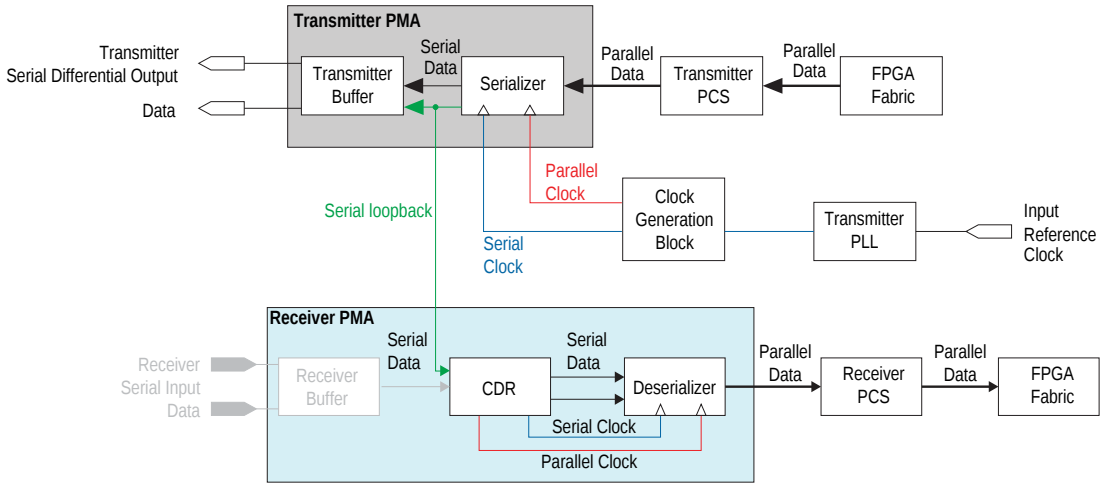


Figure 5-18: Diagnostic Loopback Path

The diagnostic loopback path sets the transmitter buffer to transmit data fed directly from the receiver buffer before being processed by the CDR circuitry. Data from the serializer is ignored by the transmitter buffer.

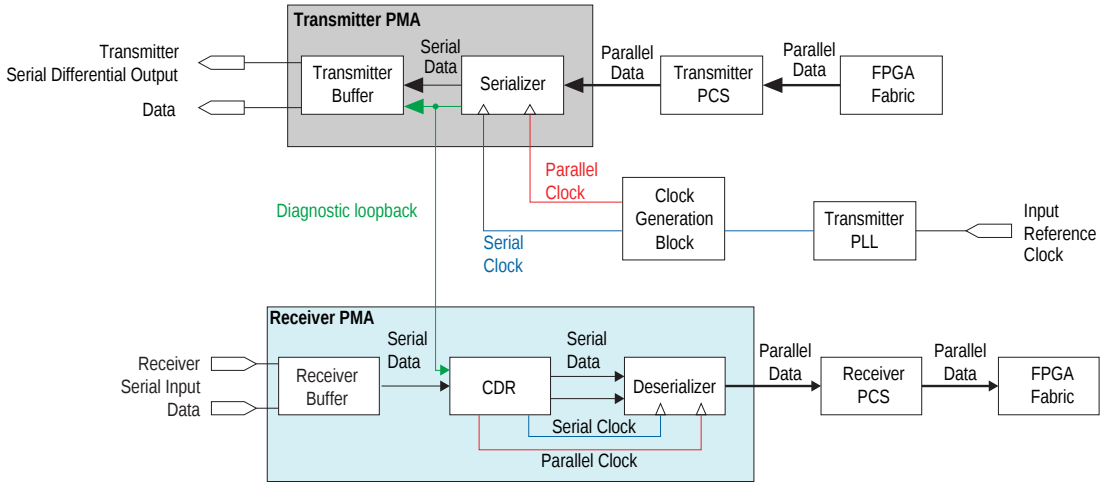
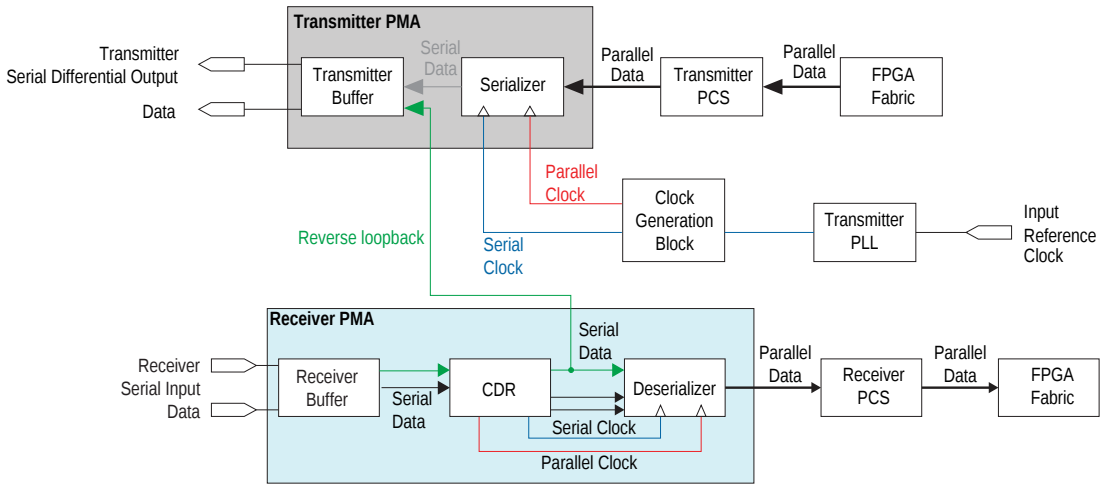


Figure 5-19: Reverse Loopback Path

The reverse loopback path sets the transmitter buffer to transmit data fed directly from the CDR recovered data. Data from the serializer is ignored by the transmitter buffer.



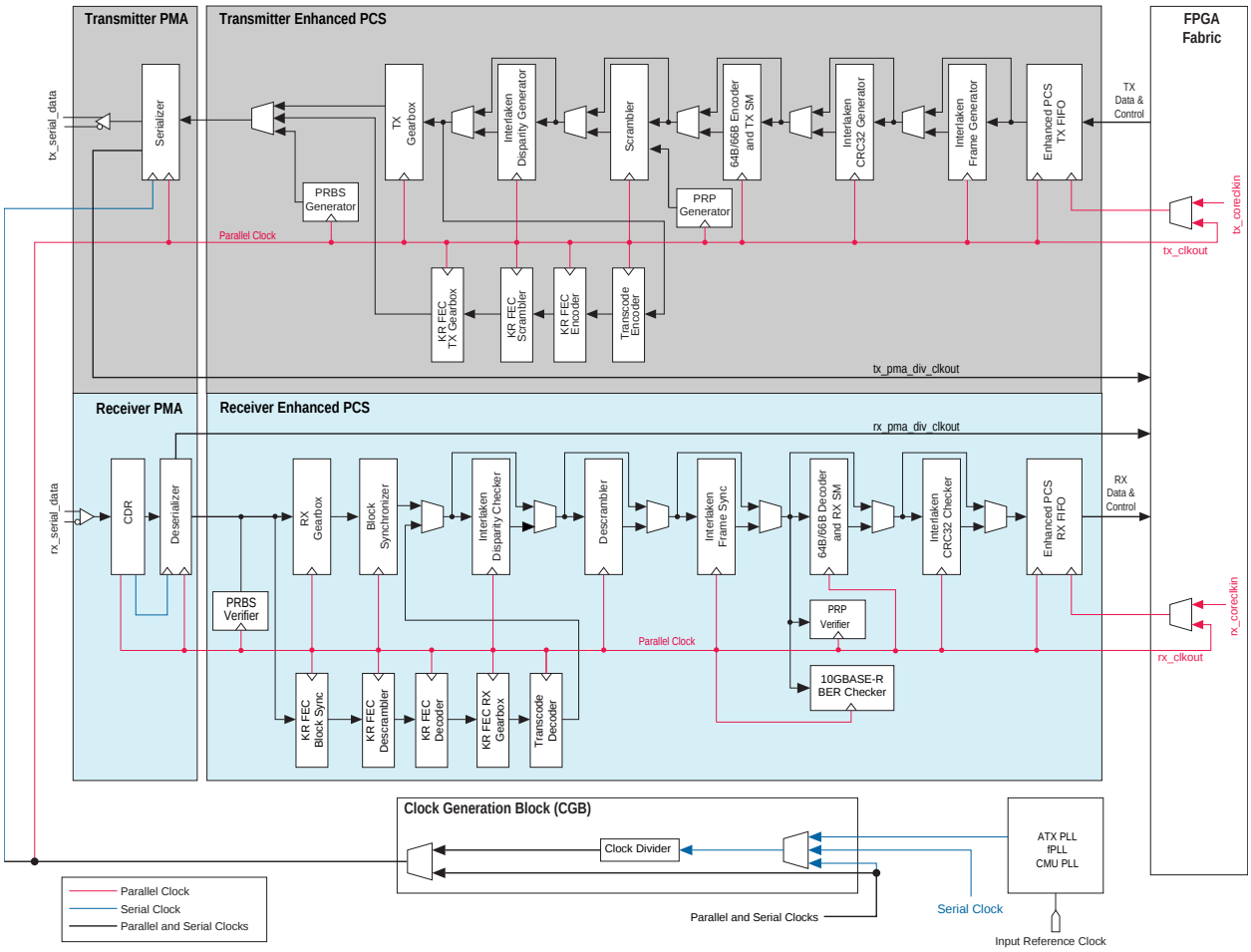
Arria 10 Enhanced PCS Architecture

You can use the Enhanced PCS to implement multiple protocols that operate at around 10 Gbps or higher line rates.

The Enhanced PCS provides the following functions:

- Performs functions common to most serial data industry standards, such as word alignment, encoding/decoding, and framing, before data is sent or received off-chip through the PMA
- Handles data transfer to and from the FPGA fabric
- Internally handles data transfer to and from the PMA
- Provides frequency compensation
- Performs channel bonding for multi-channel low skew applications

Figure 5-20: Enhanced PCS Datapath Diagram



Related Information

Implementing Protocols in Arria 10 Transceivers on page 2-1

Transmitter Datapath

Enhanced PCS TX FIFO (Shared with Standard PCS and PCIe Gen3 PCS)

The Enhanced PCS TX FIFO provides an interface between the transmitter channel PCS and the FPGA fabric. The TX FIFO can operate for phase compensation between the channel PCS and FPGA fabric. You

can also use the TX FIFO as an elastic buffer to control the input data flow, using `tx_enh_data_valid`. The TX FIFO also allows channel bonding. The TX FIFO has a width of 73 bits and a depth of 16 words.

The TX FIFO partially full and empty thresholds can also be set through the Transceiver and PLL Address Map. Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for more details.

The TX FIFO supports the following operating modes:

- Phase Compensation mode
- Register mode
- Interlaken mode
- Basic mode

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 6-1

Phase Compensation Mode

In Phase Compensation mode, the TX FIFO decouples phase variations between the FPGA fabric and transceiver clock domains. In this mode, the TX FIFO compensates for the phase difference between the read and write clocks. You can use `tx_coreclkin` (FPGA fabric clock) or `tx_clkout` (TX parallel low-speed clock) to clock the write side of the TX FIFO. `tx_clkout` clocks the read side of the TX FIFO.

Note: The TX FIFO write clock frequency and read clock frequency depend on the gearbox ratio, `tx_enh_data_valid` control signal.

For example, when using the 40:40 gearbox ratio and a data rate of 10 Gbps, the `tx_clkout` frequency is 250 MHz and the `tx_coreclkin` frequency is 250 MHz. The `tx_clkout` frequency is 257.8125 MHz in 10GBASE-R mode when using the 66:40 gearbox ratio. `tx_coreclkin` must run at 156.25 MHz.

Note: Phase Compensation can also be used in double-width mode, where the FPGA fabric data width is doubled to allow the FPGA fabric clock to run at half rate. The single/double width mode is set in the Native PHY MegaWizard. Refer to the *PLLs and Clock Networks* chapter for details about the clock frequencies when using FIFO single and double width modes.

Register Mode

In Register mode, `tx_parallel_data` (data), `tx_control` (indicates whether `tx_parallel_data` is a data or control word), and `tx_enh_data_valid` (data valid) are registered at the FIFO output. The FIFO in register mode has one register stage or one parallel clock latency.

Note: You must control the `tx_enh_data_valid` signal based on the gearbox ratio to avoid overflow or underflow in the gearbox.

Interlaken Mode

In Interlaken mode, the TX FIFO operates as an elastic buffer. In this mode, there are additional signals to control the data flow into the FIFO. Therefore, the FIFO write clock frequency does not have to be the same as the read clock frequency. You control the writing to the TX FIFO with `tx_enh_data_valid` by monitoring the FIFO flags. The goal is to prevent the FIFO from becoming full or empty. On the read side, read enable is controlled by the Interlaken frame generator.

Basic Mode

In Basic mode, the TX FIFO operates as an elastic buffer, where you control the FIFO `tx_enh_data_valid` based on FIFO flags. The FIFO read enable is controlled by gearbox data valid, which is a function of gearbox input and output data width.

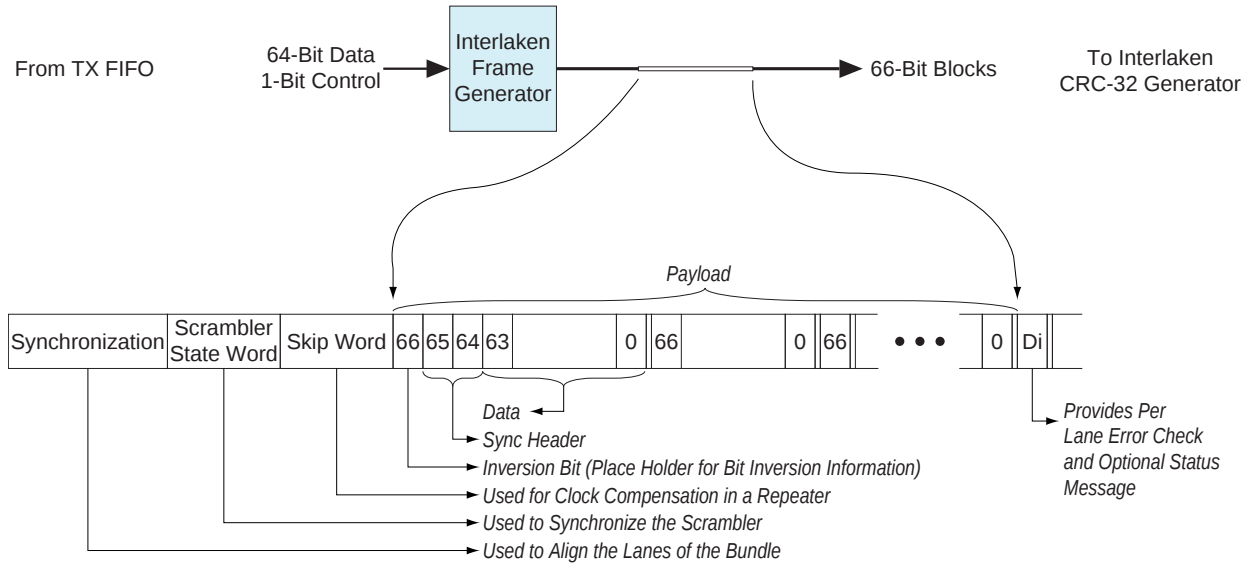
Interlaken Frame Generator

The Interlaken frame generator block takes the data from the TX FIFO and encapsulates the payload and burst/idle control words from the FPGA fabric with the framing layer's control words (Synchronization word, scrambler state word, skip word and diagnostic word) to form a metaframe. The Native PHY IP MegaWizard™ allows you to set the metaframe length from five 8-byte words to a maximum value of 8192 (64Kbyte words).

Program the same value for the metaframe length for the transmitter and receiver.

Figure 5-21: Interlaken Frame Generator

The Interlaken frame generator is available to implement the Interlaken protocol.



Interlaken CRC-32 Generator

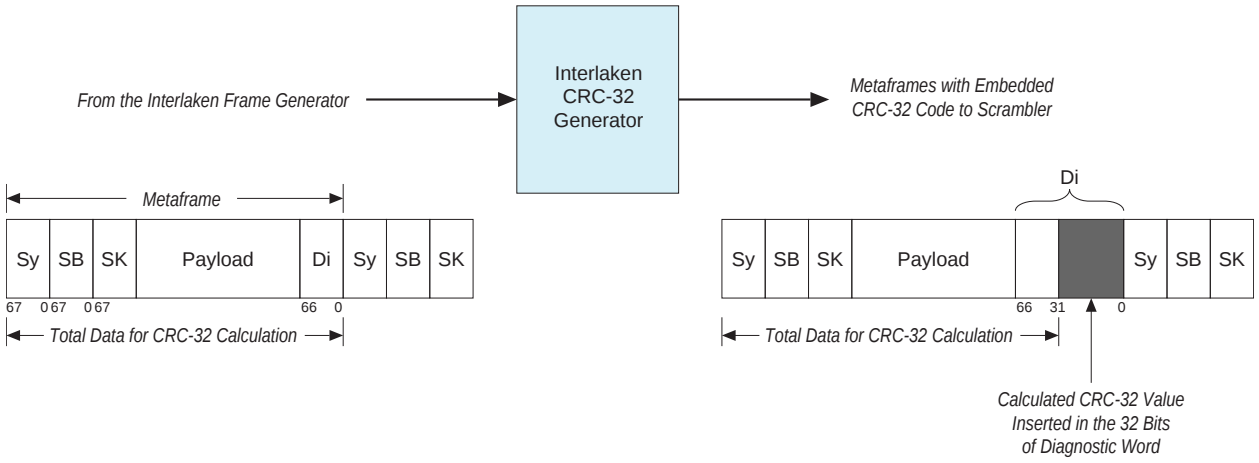
The Interlaken CRC-32 generator block receives data from the Interlaken frame generator and calculates the cyclic redundancy check (CRC) code for each block of data. This CRC code value is stored in the CRC32 field of the diagnostic word. CRC-32 provides a diagnostic tool for each lane. This helps to trace the errors on the interface back to an individual lane.

The CRC-32 calculation covers most of the metaframe, including the diagnostic word, except the following:

- Bits [66:64] of each word
- 58-bit scrambler state within the scrambler state word
- 32-bit CRC-32 field within the diagnostic word

Figure 5-22: Interlaken CRC-32 Generator

The Interlaken CRC-32 generator is available to implement the Interlaken protocol.



64B/66B Encoder and Transmitter State Machine

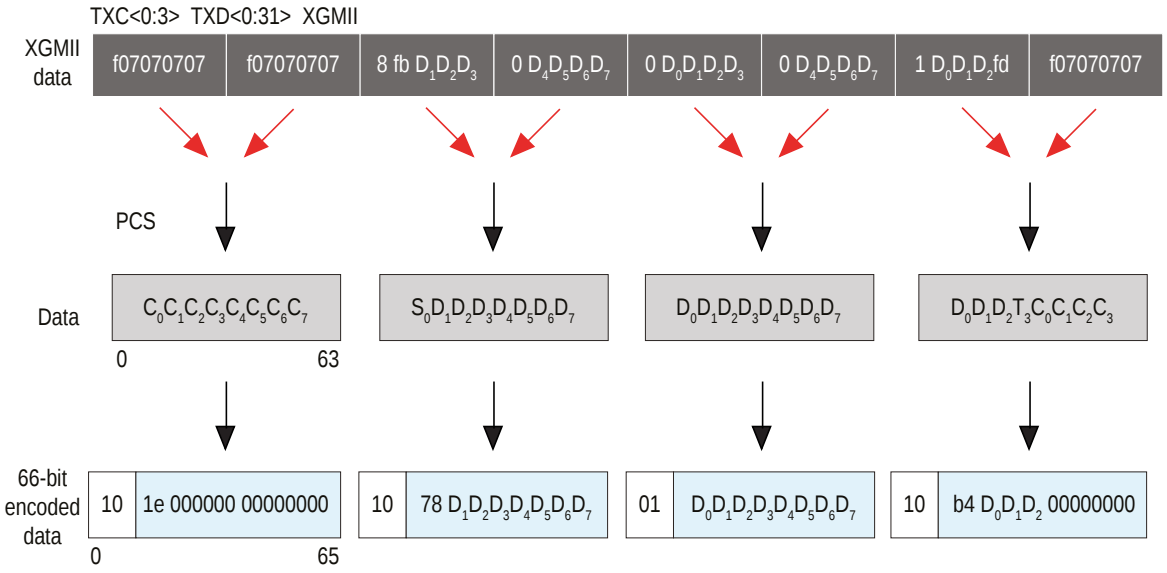
The 64B/66B encoder is used to achieve DC-balance and sufficient data transitions for clock recovery. It encodes 64-bit XGMII data and 8-bit XGMII control into 10GBASE-R 66-bit control or data blocks in accordance with Clause 49 of the IEEE802.3-2008 specification.

The 66-bit encoded data contains two overhead sync header bits that the receiver PCS uses for block synchronization and bit-error rate (BER) monitoring. The sync header is 01 for data blocks and 10 for control blocks. Sync headers are not scrambled and are used for block synchronization. (The sync headers 00 and 11 are not used, and generate an error if seen.) The remainder of the block contains the payload. The payload is scrambled and the sync header bypasses the scrambler.

The encoder block also has a state machine (TX SM) designed in accordance with the IEEE802.3-2008 specification. The TX SM ensures valid packet construction on data sent from the MAC layer. It also performs functions such as transmitting local faults under reset, as well as transmitting error codes when the 10GBASE-R PCS rules are violated.

Note: The 64B/66B encoder is available to implement the 10GBASE-R protocol.

Figure 5-23: Example Data Pattern for 64B/66B Encoding



Enhanced PCS Pattern Generators

The Enhanced PCS supports pseudo-random bit sequence (PRBS) pattern and square wave pattern generators. The pattern generators or verifiers in the Enhanced PCS are enabled by writing to the respective register bits of the Transceiver and PLL Address Map.

Refer to the Transceiver and PLL Address Map for configuration details.

PRBS Generator

The PRBS generator block generates PRBS patterns and square wave patterns.

PRBS test patterns may be considered equivalent to "noise." Use these pattern generators to test the transceiver link with a noisy signal (using the test patterns listed below) by placing the transceiver in loopback mode.

The PRBS generator supports a 64-bit PCS-PMA interface. PRBS9 supports 64-bit and 10-bit PCS-PMA interface widths to allow testing at lower data rates.

The following PRBS test patterns are supported:

- PRBS9: $x^9 + x^5 + 1$
- PRBS15: $x^{15} + x^{14} + 1$
- PRBS23: $x^{23} + x^{18} + 1$
- PRBS31: $x^{31} + x^{28} + 1$

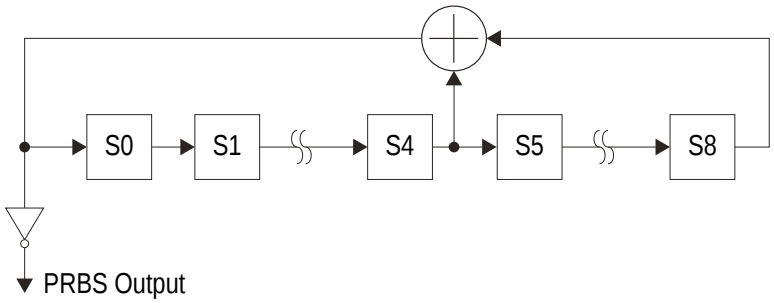
Enable the PRBS generator and select a test pattern through the reconfiguration interface.

Use PRBS9 to test transceiver links with linear impairments, and with 8B/10B.

Use PRBS15 for jitter evaluation.

Use PRBS23 or PRBS31 for jitter evaluation (data-dependent jitter) of non-8B/10B links, such as SDH/SONET/OTN jitter testers. Most 40G, 100G, and 10G applications use PRBS31 for link evaluation.

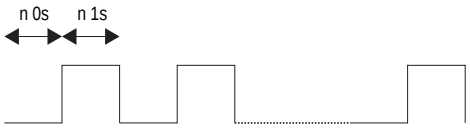
Figure 5-24: PRBS Generator for Serial Implementation of PRBS9 Pattern



Note: All supported PRBS generators are similar to the PRBS9 generator.

The square-wave generator supports 64-bit PCS-PMA interface widths. It has a programmable *n*-number of consecutive serial bit 1s and 0s, where *n* is 4, 6, or 8 (*n* defaults to 4).

Figure 5-25: Generator for Square Wave Pattern



Program the value of *n* through the Transceiver and PLL Address Map.

Pseudo-Random Pattern Generator

The pseudo-random pattern (PRP) generator is specifically designed for the 10GBASE-R and 1588 protocols. The PRP generator block operates in conjunction with the scrambler to generate pseudo-random patterns for the TX and RX tests in the 10G Ethernet mode. It generates various test patterns from various seeds loaded to the scrambler and select data patterns.

Set the following PRP generator options through the Transceiver and PLL Address Map:

- The data pattern select bit switch can be toggled between two data patterns
- The value of Seed A and Seed B can be changed

Note: You cannot enable the PRP and PRBS generators at the same time.

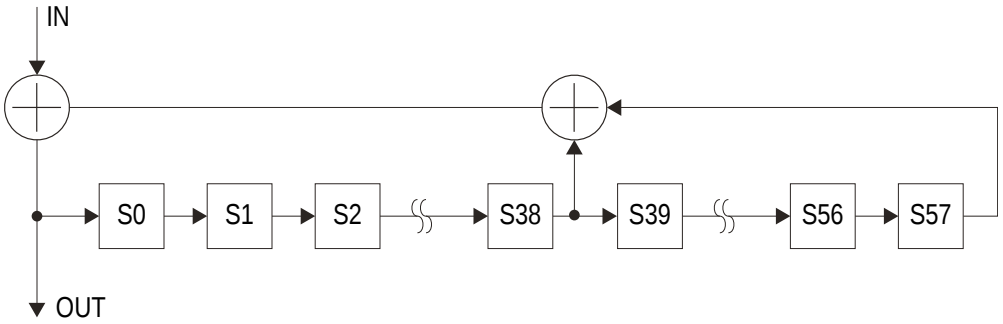
Refer to the Transceiver and PLL Address Map for configuration details.

Scrambler

The scrambler randomizes data to create transitions to DC-balance the signal and help CDR circuits. The scrambler uses a $x^{58} + x^{39} + 1$ polynomial and supports both synchronous scrambling used for Interlaken and asynchronous (also called self-synchronized) scrambling used for the 10GBASE-R protocol.

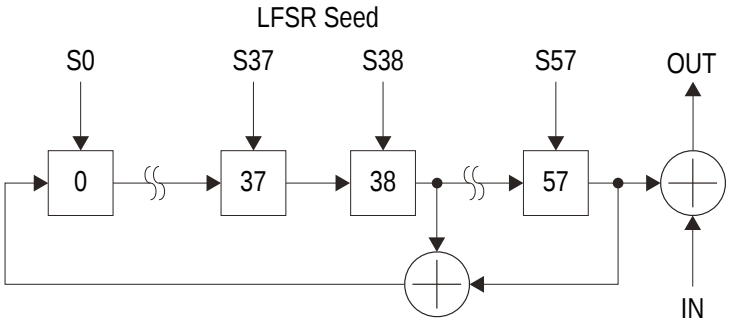
The asynchronous (self-synchronizing) mode does not require an initialization seed. Except for the two sync header bits in each 66-bit data block, the entire 64-bit payload is scrambled by feeding it into a linear feedback shift register (LFSR) continuously to generate scrambled data while the sync-header bits bypass the scrambler. The initial seed is set to all 1s. You can change the seed for the 10GBASE-R protocol using the Native PHY IP MegaWizard.

Figure 5-26: Asynchronous Scrambler in Serial Implementation



In synchronous mode, the scrambler is initially reset to different programmable seeds on each lane. The scrambler then runs by itself. Its current state is XOR'd with the data to generate scrambled data. A data checker in the scrambler monitors the data to determine if it should be scrambled or not. If a synchronization word is found, it is transmitted without scrambling. If a Scrambler State Word is detected, the current scramble state is written into the 58-bit scramble state field in the Scrambler State Word and sent over the link. The receiver uses this scramble state to synchronize the descrambler. The seed is automatically set for Interlaken protocol.

Figure 5-27: Synchronous Scrambler Showing Different Programmable Seeds



Interlaken Disparity Generator

The Interlaken disparity generator block is in accordance with the Interlaken protocol specification and provides a DC-balanced data output.

The Interlaken protocol solves the unbounded baseline wander, or DC imbalance, of the 64B/66B coding scheme used in 10Gb Ethernet by inverting the transmitted data. The disparity generator monitors the transmitted data and makes sure that the running disparity always stays within a ±96-bit bound. It adds the 67th bit (bit 66) to signal the receiver whether the data is inverted or not.

Table 5-3: Inversion Bit Definition

Bit 66	Interpretation
0	Bits [63:0] are not inverted; the receiver processes this word without modification
1	Bits [63:0] are inverted; the receiver inverts the bits before processing this word

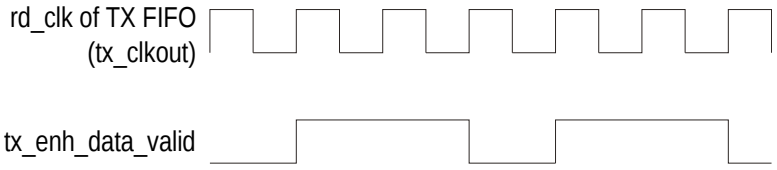
Note: The Interlaken disparity generator is available to implement the Interlaken protocol.

TX Gearbox, TX Bitslip and Polarity Inversion

The TX gearbox adapts the PCS data width to the smaller bus width of the PCS-PMA interface (Gearbox Reduction). It supports different ratios (FPGA fabric-PCS Interface Width: PCS-PMA Interface Width) such as 66:32, 66:40, 67:32, 67:40, 50:40, 64:32, 64:40, 40:40, 32:32, 64:64, 67:64, and 66:64. The gearbox mux selects a group of consecutive bits from the input data bus depending on the gearbox ratio and the data valid control signals.

Data valid generation logic is essential for gearbox operation. Each block of data is accompanied by `tx_enh_data_valid` (data valid signal) which "qualifies" the block as valid or not. The data valid toggling pattern is dependent on the data width conversion ratio. For example, if the ratio is 66:40, the data valid signal is high in 20 out of 33 cycles or approximately 2 out of 3 cycles and the pattern repeats every 33 `tx_clkout` (TX low-speed parallel clock) cycles.

Figure 5-28: 66:40 Data Valid Pattern



The TX gearbox also has a bit slipping feature to adjust the data skew between channels. The TX parallel data is slipped on the rising edge of `tx_enh_bitslip` before it is passed to the PMA. The maximum number of the supported bitslips is PCS data width-1 and the slip direction is from MSB to LSB and from current to previous word.

Figure 5-29: TX Bitslip

`tx_enh_bitslip` = 2 and PCS width of gearbox is 67

	Current Word							Previous Word						
Bit Index	66	65	64	...	2	1	0	66	65	64	...	2	1	0

You can use transmitter data polarity inversion to invert the polarity of every bit of the input data word to the serializer in the transmitter path. The inversion has the same effect as swapping the positive and negative signals of the differential TX buffer. This is useful if these signals are reversed on the board or backplane layout. Enable polarity inversion through the Native PHY IP MegaWizard.

KR FEC Blocks

The KR FEC blocks in the Enhanced PCS are designed in accordance with the 10G-KRFEC and 40G-KRFEC of the IEEE 802.3 specification. The KR FEC implements the Forward Error Correction (FEC) sublayer, a sublayer between the PCS and PMA sublayers.

Most data transmission systems, such as Ethernet, have minimum requirements for the bit error rate (BER). However, due to channel distortion or noise in the channel, the required BER may not be achievable. In these cases, adding a forward error control correction can improve the BER performance of the system.

The FEC sublayer is optional and can be bypassed. When used, it can provide additional margin to allow for variations in manufacturing and environmental conditions. FEC can achieve the following objectives:

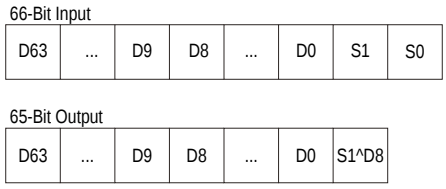
- Support a forward error correction mechanism for the 10GBASE-R/KR and 40GBASE-R/KR protocols
- Support the full duplex mode of operation of the Ethernet MAC
- Support the PCS, PMA, and Physical Medium Dependent (PMD) sublayers defined for the 10GBASE-R/KR and 40GBASE-R/KR protocols

With KR FEC, the BER performance of the system can be improved.

Transcode Encoder

The KR forward error correction (KR FEC) transcode encoder block performs the 64B/66B to 65-bit transcoder function by generating the transcode bit. The transcode bit is generated from a combination of 66 bits after the 64B/66B encoder which consists of a 2-bit synchronization header (S0 and S1) and a 64-bit payload (D0, D1,…, D63). To ensure a DC-balanced pattern, the transcode word is generated by performing an XOR function on the second synchronization bit S1 and payload bit D8. The transcode bit becomes the LSB of the 65-bit pattern output of the transcode encoder.

Figure 5-30: Transcode Encoder



KR FEC Encoder

FEC (2112,2080) is an FEC code specified in Clause 74 of the IEEE 802.3 specification. The code is a shortened cyclic code (2112, 2080). For each block of 2080 message bits, another 32 parity checks are generated by the encoder to form a total of 2112 bits. The generator polynomial is:

$g(x) = x^{32} + x^{23} + x^{21} + x^{11} + x^{2} + 1$

KR FEC Scrambler

The KR FEC scrambler block performs scrambling based on the generation polynomial $x^{58} + x^{39} + 1$, which is necessary for establishing FEC block synchronization in the receiver and to ensure DC balance.

KR FEC TX Gearbox

The KR FEC TX gearbox converts 65-bit input words to 64-bit output words to interface the KR FEC encoder with the PMA. This gearbox is different from the TX gearbox used in the Enhanced PCS. The KR FEC TX gearbox aligns with the FEC block. Because the encoder output (also the scrambler output) has its unique word size pattern, the gearbox is specially designed to handle that pattern.

Receiver Datapath

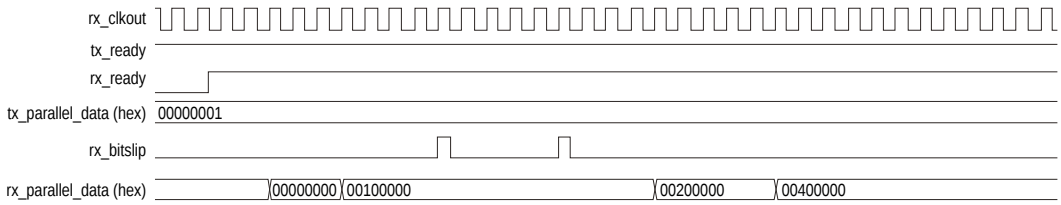
RX Gearbox, RX Bitslip, and Polarity Inversion

The RX gearbox adapts the PMA data width to the larger bus width of the PCS channel (Gearbox Expansion). It supports different ratios (PCS-PMA interface width : FPGA fabric–PCS interface width) such as 32:66, 40:66, 32:67, 40:67, 40:50, 32:64, 40:64, 40:40, 32:32, 64:64, 67:64, and 66:64 and a bit slipping feature.

RX bitslip is engaged when the RX block synchronizer or `rx_bitslip` is enabled to shift the word boundary. On the rising edge of the bitslip signal of the RX block synchronizer or `rx_bitslip` from the FPGA fabric, the word boundary is shifted by 1 bit. Each bit slip removes the earliest received bit from the received data.

Figure 5-31: RX Bitslip

`rx_bitslip` is toggled two times, which shifts the `rx_parallel_data` boundary two bits.



The receiver gearbox can invert the polarity of the incoming data. This is useful if the receiver signals are reversed on the board or backplane layout. Enable polarity inversion through the Native PHY IP MegaWizard.

Block Synchronizer

The block synchronizer determines the block boundary of a 66-bit word in the case of the 10GBASE-R protocol or a 67-bit word in the case of the Interlaken protocol. The incoming data stream is slipped one bit at a time until a valid synchronization header (bits 65 and 66) is detected in the received data stream. After the predefined number of synchronization headers (as required by the protocol specification) is detected, the block synchronizer asserts `rx_enh_blk_lock` (block lock status signal) to other receiver PCS blocks down the receiver datapath and to the FPGA fabric.

Note: The block synchronizer is designed in accordance with Interlaken Protocol specification (as described in Figure 13 of Interlaken Protocol Definition v1.2) and 10GBASE-R protocol specification (as described in IEEE 802.3-2008 clause-49).

Interlaken Disparity Checker

The Interlaken disparity checker examines the received inversion bit inserted by the far end disparity generator, to determine whether to reverse the inversion process of the Interlaken disparity generation.

Note: The Interlaken disparity checker is available to implement the Interlaken protocol.

Descrambler

The descrambler block descrambles received data to regenerate unscrambled data using the $x^{58} + x^{39} + 1$ polynomial. Like the scrambler, it operates in asynchronous mode or synchronous mode.

Related Information

Scrambler on page 5-20

Interlaken Frame Synchronizer

The Interlaken frame synchronizer delineates the metaframe boundaries and searches for each of the framing layer control words: Synchronization, Scrambler State, Skip, and Diagnostic. When four consecutive synchronization words have been identified, the frame synchronizer achieves the frame locked state. Subsequent metaframes are then checked for valid synchronization and scrambler state words. If four consecutive invalid synchronization words or three consecutive mismatched scrambler state words are

received, the frame synchronizer loses frame lock. In addition, the frame synchronizer provides `rx_enh_frame_lock` (receiver metaframe lock status) to the FPGA fabric.

Note: The Interlaken frame synchronizer is available to implement the Interlaken protocol.

64B/66B Decoder and Receiver State Machine

The 64B/66B decoder reverses the 64B/66B encoding process. The decoder block also contains a state machine (RX SM) designed in accordance with the IEEE802.3-2008 specification. The RX SM checks for a valid packet structure in the data sent from the remote side. It also performs functions such as sending local faults to the Media Access Control (MAC)/Reconciliation Sublayer (RS) under reset and substituting error codes when the 10GBASE-R and 10GBASE-KR PCS rules are violated.

Note: The 64B/66B decoder is available to implement the 10GBASE-R protocol.

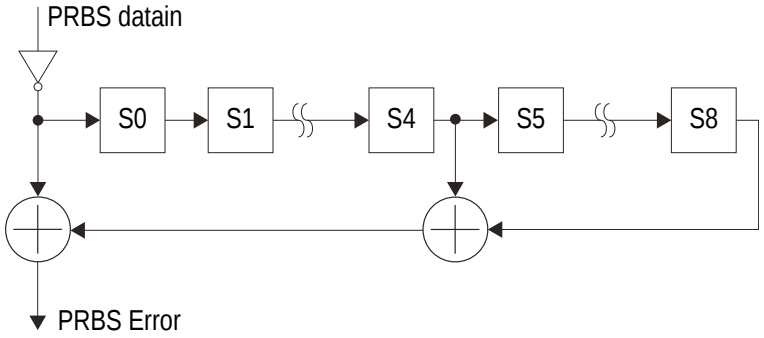
PRBS Verifier

The pseudo-random bit stream (PRBS) block verifies the pattern generated by the PRBS generator. The verifier supports the 64-bit PCS-PMA interface. PRBS9 also supports 10-bit PMA data width to allow testing at a lower data rate.

The following PRBS verifiers are supported:

- PRBS9: $x^9 + x^5 + 1$
- PRBS15: $x^{15} + x^{14} + 1$
- PRBS23: $x^{23} + x^{18} + 1$
- PRBS31: $x^{31} + x^{28} + 1$

Figure 5-32: PRBS9 Verify Serial Implementation



The PRBS verifier has the following control and status signals available to the FPGA fabric:

- `rx_prbs_done`—Indicates the PRBS sequence has completed one full cycle. It stays high until you reset it with `rx_prbs_err_clr`.
- `rx_prbs_err`—Goes high if an error occurs. This signal is pulse-extended to allow you to capture it in the RX FPGA CLK domain.
- `rx_prbs_err_clr`—Used to reset the `rx_prbs_err` signal.

The Enhanced RX datapath does not include a verifier for the square wave.

Enable the PRBS verifier control and status ports through the Native PHY IP MegaWizard in the Quartus II software. The PRBS pattern is automatically selected to match the PRBS generator pattern that you selected. The Transceiver and PLL Address Map can be used to enable or disable PRBS verifier.

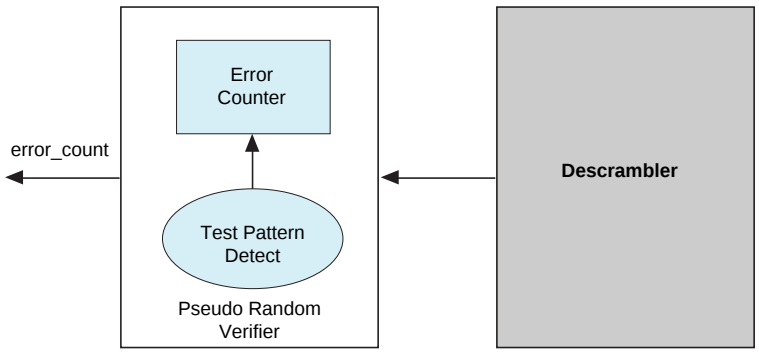
PRP Verifier

The Pseudo Random Pattern (PRP) verifier is available for 10GBASE-R and 10GBASE-R 1588 protocol modes. The PRP verifier monitors the output of the descrambler when block synchronization is achieved.

The PRP verifier:

- Searches for a test pattern (two local faults, or all 0s) or its inverse
- Tracks the number of mismatches with a 16-bit error counter

Figure 5-33: PRP Verifier



Note: You cannot enable the PRP and PRBS verifiers at the same time.

Enable the PRP verifier through the Transceiver and PLL Address Map. The PRP pattern is automatically selected to match the PRP generator pattern that you selected.

10GBASE-R Bit-Error Rate (BER) Checker

The 10GBASE-R BER checker block is designed in accordance with the 10GBASE-R protocol specification as described in IEEE 802.3-2008 clause-49. After block lock synchronization is achieved, the BER checker starts to count the number of invalid synchronization headers within a 125-µs period. If more than 16 invalid synchronization headers are observed in a 125-µs period, the BER checker provides the status signal `rx_enh_highber` to the FPGA fabric, indicating a high bit error rate condition.

When the optional control input `rx_enh_highber_clr_cnt` is asserted, the internal counter for the number of times the BER state machine has entered the "BER_BAD_SH" state is cleared.

When the optional control input `rx_enh_clr_errblk_count` is asserted, the internal counter for the number of times the RX state machine has entered the "RX_E" state for the 10GBASE-R protocol is cleared. In modes where the FEC block in enabled, the assertion of this signal resets the status counters within the RX FEC block.

Note: The 10GBASE-R BER checker is available to implement the 10GBASE-R protocol.

Interlaken CRC-32 Checker

The Interlaken CRC-32 checker verifies that the data transmitted has not been corrupted between the transmit PCS and the receive PCS. The CRC-32 checker calculates the 32-bit CRC for the received data and compares it against the CRC value that is transmitted within the diagnostic word. `rx_enh_crc32_err` (CRC error signal) is sent to the FPGA fabric.

Enhanced PCS RX FIFO

The Enhanced PCS RX FIFO is designed to compensate for the phase and/or clock difference between the receiver channel PCS and the FPGA fabric. It can operate as a phase-compensation, clock-compensation, elastic buffer, or a deskew FIFO in Interlaken mode. The RX FIFO has a width of 74 bits and a depth of 32 words for all protocols.

The RX FIFO supports the following modes:

- Phase Compensation mode
- Register mode
- Interlaken mode (deskew FIFO)
- 10GBASE-R mode (clock compensation FIFO)
- Basic mode (elastic buffer FIFO)

Phase Compensation Mode

The RX FIFO compensates for the phase difference between the read clock and write clocks. `rx_clkout` (RX parallel low-speed clock) clocks the write side of the RX FIFO. `rx_coreclkin` (FPGA fabric clock) or `rx_clkout` clocks the read side of the RX FIFO.

When phase compensation is used in double-width mode, the FPGA data width is doubled to allow the FPGA fabric clock to run at half rate, similar to the TX FIFO phase compensation in double-width mode.

Register Mode

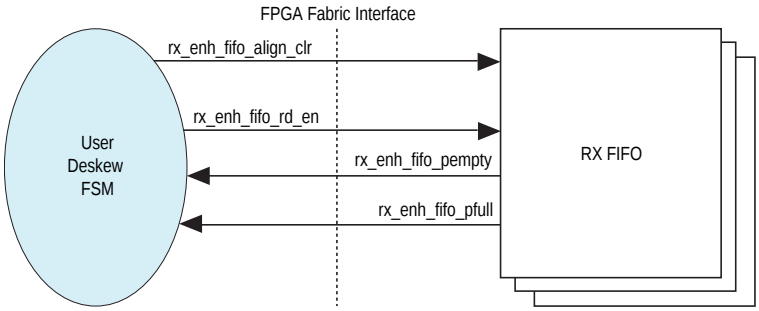
In Register mode, `rx_parallel_data` (data), `rx_control` (indicates whether `rx_parallel_data` is a data or control word), and `rx_enh_data_valid` (data valid) are registered at the FIFO output. The RX FIFO in register mode has one register stage or one parallel clock latency.

Interlaken Mode

In Interlaken mode, the RX FIFO operates as an Interlaken deskew FIFO. To implement the deskew process, implement an FSM that controls the FIFO operation based on available FPGA input and output flags.

For example, after frame lock is achieved, data is written after the first alignment word (SYNC word) is found on that channel. As a result, `rx_enh_fifo_pempty` (FIFO partially empty flag) of that channel goes low. You must monitor the `rx_enh_fifo_pempty` and `rx_enh_fifo_pfull` flags of all channels. If `rx_enh_fifo_pempty` flags from all channels deassert before any `rx_enh_fifo_pfull` flag asserts, which implies alignment word has been found on all lanes of the link, you start reading from all the FIFOs by asserting `rx_enh_fifo_rd_en`. Otherwise, if a `rx_enh_fifo_pfull` flag from any channel goes high before a `rx_enh_fifo_pempty` flag deassertion on all channels, you must reset the FIFO by toggling the `rx_enh_fifo_align_clr` signal and repeating the process.

Figure 5-34: RX FIFO as Interlaken Deskew FIFO



10GBASE-R Mode

In 10GBASE-R mode, the RX FIFO operates as a clock compensation FIFO. When the block synchronizer achieves block lock, data is sent through the FIFO. Idles/Ordered Sets (OS) are deleted and Idles are inserted to compensate for the clock difference between the RX low-speed parallel clock and the FPGA fabric clock (±100 ppm for a maximum packet length of 64,000 bytes).

Idle/OS Deletion

Deletion of Idles occurs in groups of four OS (when there are two consecutive OS) until the `rx_enh_fifo_rd_pempty` flag deasserts. Every word—consisting of a lower word (LW) and an upper word (UW)—is checked for whether it can be deleted by looking at both the current and previous words.

For example, the current LW can be deleted if it is Idle and the previous UW is not a Terminate.

Table 5-4: Conditions Under Which a Word Can be Deleted

In this table X=don't care, T=Terminate, I=Idle, and OS=order set.

Deletable	Case	Word	Previous	Current	Output	
Lower Word	1	UW	!T	X	!T	X
		LW	X	I	X	X
	2	UW	OS	X	OS	X
		LW	X	OS	X	X
Upper Word	1	UW	X	I	X	X
		LW	X	!T	X	!T
	2	UW	X	OS	X	X
		LW	X	OS	X	OS

If only one word is deleted, data shifting is necessary because the datapath is two words wide. After two words have been deleted, the FIFO stops writing for one cycle and a synchronous flag (`rx_control[8]`) appears on the next block of 8-byte data. There is also an asynchronous status signal `rx_enh_fifo_del`, which does not go through the FIFO.

Figure 5-35: IDLE Word Deletion

This figure shows the deletion of IDLE words from the receiver data stream.

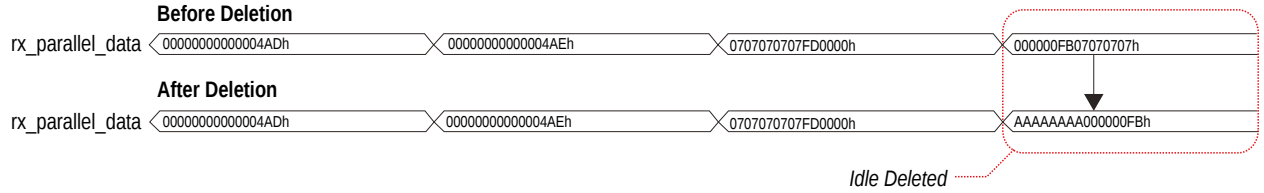
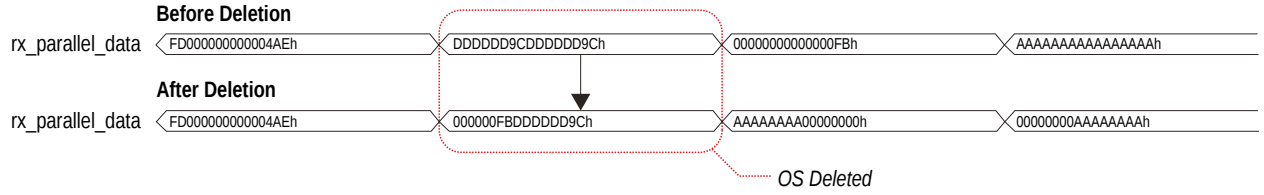


Figure 5-36: OS Word Deletion

This figure shows the deletion of Ordered set words in the receiver data stream.



Idle Insertion

Idle insertion occurs in groups of 8 Idles when the `rx_enh_fifo_pempty` flag is deasserted. Idles can be inserted following Idles or OS. Idles are inserted in groups of 8 bytes. Data shifting is not necessary. There is a synchronous status `rx_enh_fifo_insert` signal that is attached to the 8-byte Idles being inserted.

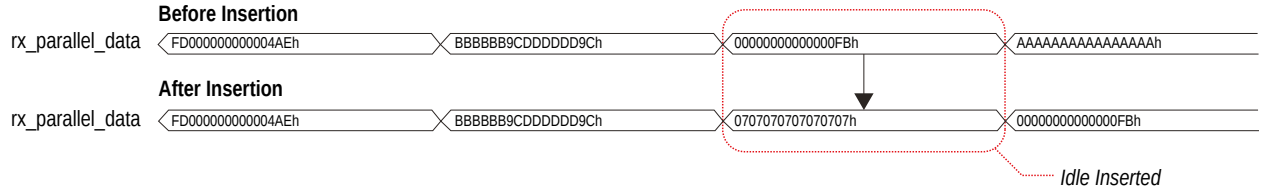
Table 5-5: Cases Where Two Idle Words are Inserted

In this table X=don't care, S=start, OS=order set, I-DS=idle in data stream, and I-In=idle inserted. In cases 3 and 4, the Idles are inserted between the LW and UW.

Case	Word	Input	Output	
1	UW	I-DS	I-DS	I-In
	LW	X	X	I-In
2	UW	OS	OS	I-In
	LW	X	X	I-In
3	UW	S	I-In	S
	LW	I-DS	I-DS	I-In
4	UW	S	I-In	S
	LW	OS	OS	I-In

Figure 5-37: IDLE Word Insertion

This figure shows the insertion of IDLE words in the receiver data stream.



Basic Mode

In Basic mode, the RX FIFO operates as an elastic buffer. The FIFO write enable is controlled by gearbox data valid, which is a function of gearbox input and output data width. You can monitor the `rx_enh_fifo_pempty` and `rx_enh_fifo_pfull` flags to determine whether to read from the FIFO or not.

RX KR FEC Blocks

KR FEC Block Synchronization

You can obtain FEC block delineation for the RX KR FEC by locking onto correctly received FEC blocks with the KR FEC block synchronization.

Note: The KR FEC block synchronization is available to implement the 10GBASE-KR protocol.

KR FEC Descrambler

The KR FEC descrambler block descrambles received data to regenerate unscrambled data using the $x^{58} + x^{39} + 1$ polynomial. Before the block boundary in the KR FEC sync block is detected, the data at the input of the descrambler is sent directly to the KR FEC decoder. When the boundary is detected, the aligned word from the KR FEC sync block is descrambled with the Psuedo Noise (PN) sequence and then sent to the KR FEC decoder.

KR FEC Decoder

The KR FEC decoder block performs the FEC (2112, 2080) decoding function by analyzing the received 32 65-bit blocks for errors. It can correct burst errors of 11 bits or less per FEC block.

KR FEC RX Gearbox

The KR FEC RX gearbox block adapts the PMA data width to the larger bus width of the PCS channel. It supports a 64:65 ratio.

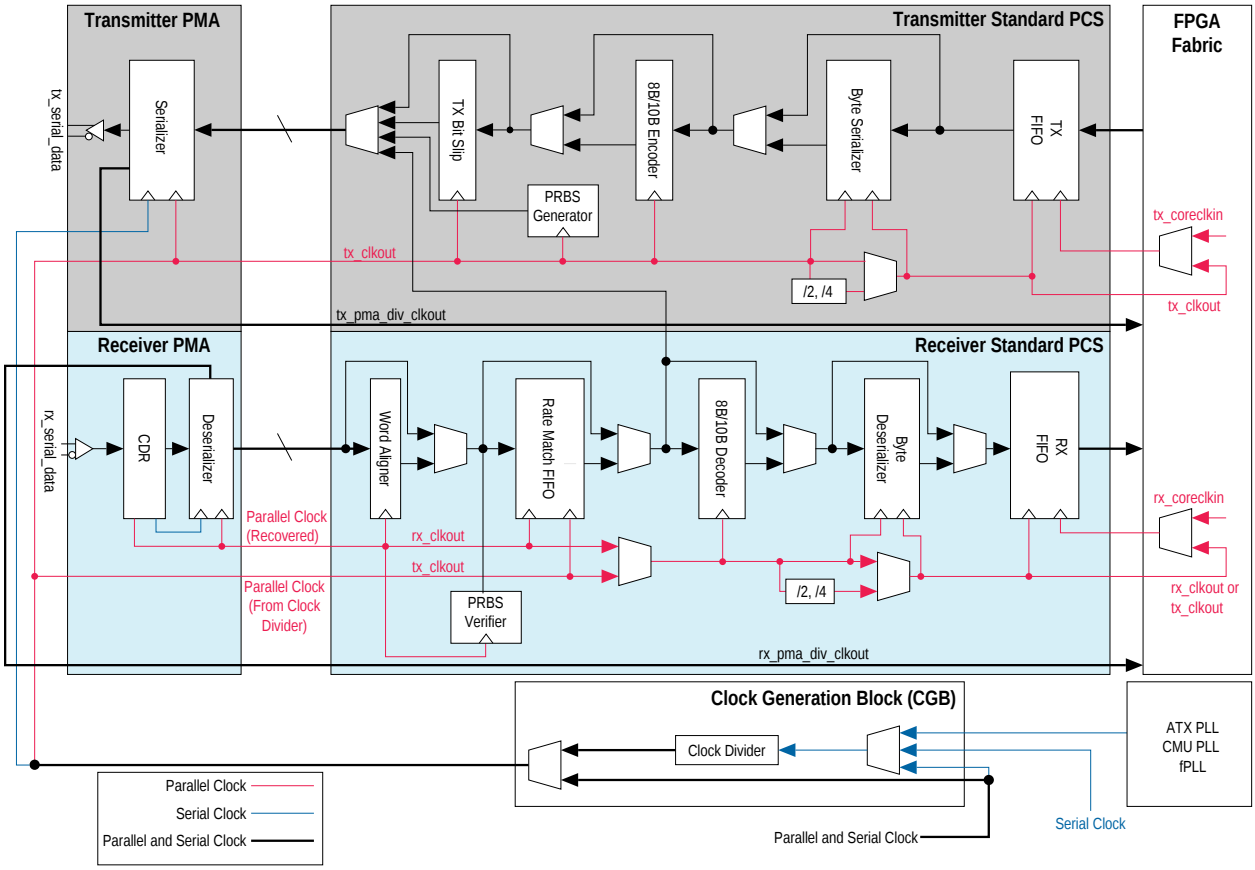
Transcode Decoder

The transcode decoder block performs the 65-bit to 64B/66B reconstruction function by regenerating the 64B/66B synchronization header.

Arria 10 Standard PCS Architecture

The standard PCS can operate at a data rate up to 12.5 Gbps. Protocols such as PCI-Express, CPRI 4.2+, GigE, IEEE 1588 are supported in Hard PCS while the other protocols can be implemented using Basic/Custom (Standard PCS) transceiver configuration rules.

Figure 5-38: Standard PCS Datapath Diagram

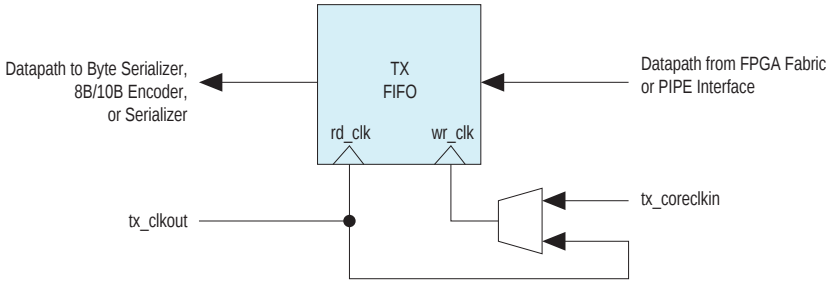


Transmitter Datapath

TX FIFO (Shared with Enhanced PCS and PCIe Gen3 PCS)

The TX FIFO interfaces between the transmitter PCS and the FPGA fabric and ensures reliable transfer of data and status signals. It compensates for the phase difference between the FPGA fabric clock and `tx_clkout` (the low-speed parallel clock). The TX FIFO operates in low latency mode and register mode.

Figure 5-39: TX FIFO Block Diagram



The TX FIFO read port is clocked by the low speed parallel clock and its write port is clocked by either `tx_clkout` or `tx_coreclkin`. `tx_clkout` is used when only one channel is being used. `tx_coreclkin` is used when using multiple channels. The TX FIFO is shared with PCIe Gen3 and Enhanced PCS data paths.

TX FIFO Low Latency Mode

The low latency mode incurs two to three cycles of latency (latency uncertainty) when connecting it with the FPGA fabric. The FIFO empty and the FIFO full threshold values are made closer so that the depth of the FIFO decreases, which in turn decreases the latency.

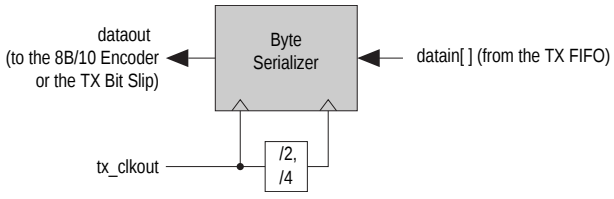
TX FIFO Register Mode

The register mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock. The register mode incurs only one clock cycle of latency when interfacing to the FPGA fabric.

Byte Serializer

In certain applications, the FPGA fabric interface cannot operate at the same clock rate as the transmitter channel (PCS) because the transmitter channel is capable of operating at higher clock rates compared to the FPGA fabric. The byte serializer allows the transmitter channel to operate at higher data rates while keeping the FPGA fabric interface clock rate below its maximum limit. This is accomplished by decreasing the channel width two or four times (FPGA fabric-to-PCS interface width) and doubling/quadrupling the clock rate. The byte serializer is disabled, or operates in Serialize x2 or Serialize x4 modes.

Figure 5-40: Byte Serializer Block Diagram



Related Information

- **Implementing Protocols in Arria 10 Transceivers** on page 2-1
- **Resetting Transceiver Channels** on page 4-1

Bonded Byte Serializer

The bonded byte serializer is available in Arria 10, and is used in applications such as PIPE, CPRI, and custom applications where multiple channels are grouped together. The bonded byte serializer is implemented by bonding all the control signals to prevent skew induction between channels during byte serialization. In this configuration, one of the channels acts as master and the remaining channels act as slaves.

Byte Serializer Disabled Mode

In disabled mode, the byte serializer is bypassed. The data from the TX FIFO is directly transmitted to the 8B/10B encoder, TX Bitslip, or Serializer, depending on whether or not the 8B/10B encoder and TX Bitslip are enabled. Disabled mode is used in low speed applications such as GigE, where the FPGA fabric and the TX standard PCS can operate at the same clock rate.

Byte Serializer Serialize x2 Mode

The serialize x2 mode is used in high-speed applications such as the PCIe Gen1 or Gen2 protocol implementation, where the FPGA fabric cannot operate as fast as the TX PCS.

In serialize x2 mode, the byte serializer serializes 16-bit, 20-bit (when 8B/10B encoder is not enabled), 32-bit and 40-bit (when 8B/10B encoder is not enabled) input data into 8-bit, 10-bit, 16-bit and 20-bit data, respectively. As the parallel data width from the TX FIFO is halved, the clock rate is doubled.

After byte serialization, the byte serializer forwards the least significant word first followed by the most significant word. For example, if the FPGA fabric-to-PCS Interface width is 32, the byte serializer forwards `tx_parallel_data[15:0]` first, followed by `tx_parallel_data[31:16]`.

Related Information

PCI Express on page 2-179
For more information about using the Serialize x2 mode in the PCIe protocol.

Byte Serializer Serialize x4 Mode

The serialize x4 mode is used in high-speed applications such as the PCIe Gen3 protocol mode, where the FPGA fabric cannot operate as fast as the TX PCS.

In serialize x4 mode, the byte serializer serializes 32-bit data into 8-bit data. As the parallel data width from the TX FIFO is divided four times, the clock rate is quadrupled.

After byte serialization, the byte serializer forwards the least significant word first followed by the most significant word. For example, if the FPGA fabric-to-PCS Interface width is 32, the byte serializer forwards `tx_parallel_data[15:0]` first, followed by `tx_parallel_data[31:16]`.

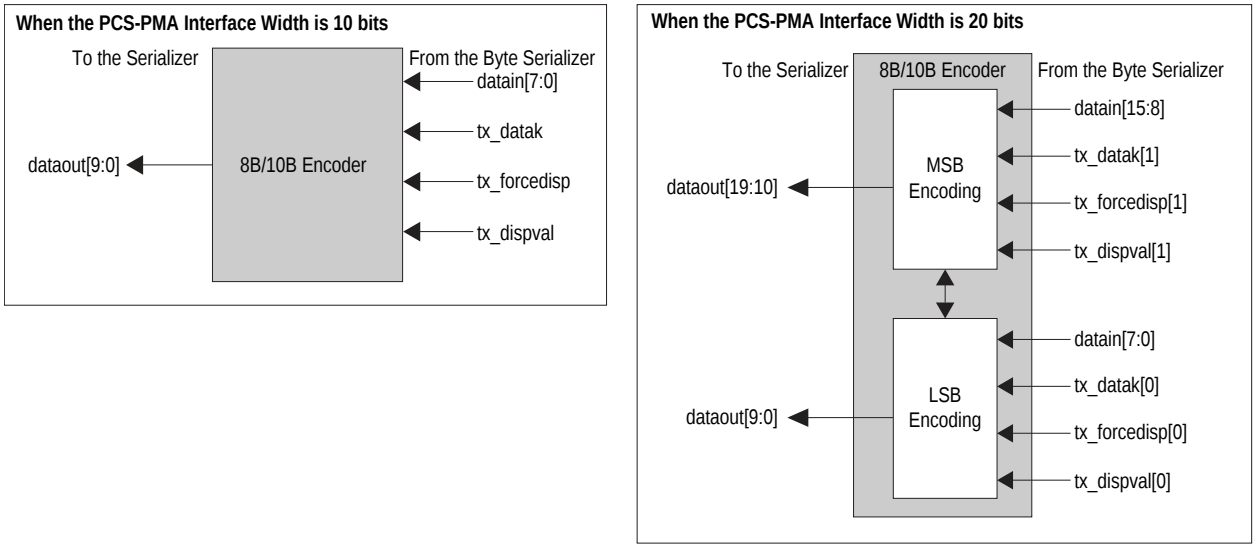
Related Information

PCI Express on page 2-179
For more information about using the Serialize x4 mode in the PCIe protocol.

8B/10B Encoder

The 8B/10B encoder takes in 8 -bit data and 1-bit control as input and converts them into a 10-bit output. The 8B/10B encoder automatically performs running disparity check for the 10-bit output. Additionally, the 8B/10B encoder can control the running disparity manually using the `tx_forcedisp` and `tx_dispval` ports.

Figure 5-41: 8B/10B Encoder Block Diagrams

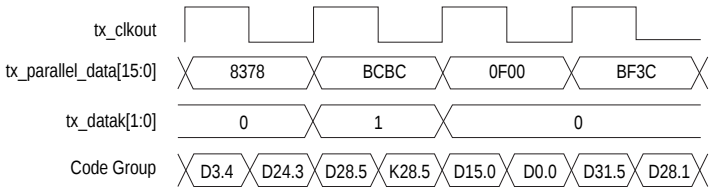


When the PCS-PMA interface width is 10 bits, one 8B/10B encoder is used to convert the 8-bit data into a 10-bit output. When the PCS-PMA interface width is 20 bits, two cascaded 8B/10B encoders are used to convert the 16-bit data into a 20-bit output, wherein the first eight bits (LSByte) is encoded by the first 8B/10B encoder and the next eight bits (MSByte) is encoded by the second 8B/10B encoder. The running disparity of the LSByte is calculated first and passed on to the second encoder to calculate the running disparity of the MSByte.

Note: You cannot enable the 8B/10B encoder when the PCS-PMA interface width is 8 bits or 16 bits.

8B/10B Encoder Control Code Encoding

Figure 5-42: Control Code Encoding Diagram



`tx_datak` is used to indicate whether the 8-bit data being sent at the `tx_parallel_data` port should be a control word or a data word. When `tx_datak` is high, the 8-bit data is encoded as a control word (Kx.y). When `tx_datak` is low, the 8-bit data is encoded as a data word (Dx.y). Depending upon the PCS-PMA interface width, the width of `tx_datak` is either 1 bit or 2 bits. When the PCS-PMA interface width is 10 bits, `tx_datak` is a 1-bit word. When the PCS-PMA interface width is 20 bits, `tx_datak` is a 2-bit word. The LSB of `tx_datak` corresponds to the LSByte of the input data sent to the 8B/10B encoder and the MSB corresponds to the MSByte of the input data sent to the 8B/10B encoder.

Related Information

Refer to *Specifications & Additional Information* for more information about 8B/10B encoder codes.

8B/10B Encoder Reset Condition

`tx_digitalreset` resets the 8B/10B encoder. During the reset condition, the 8B/10B encoder outputs K28.5 continuously until `tx_digitalreset` goes low.

8B/10B Encoder Idle Character Replacement Feature

The idle character replacement feature is used in protocols such as Gigabit Ethernet, which requires the running disparity to be maintained during idle sequences. During these idle sequences, the running disparity has to be maintained such that the first byte of the next packet always starts when the running disparity of the current packet is negative.

When an ordered set, which consists of two code-groups, is received by the 8B/10B encoder, the second code group will be converted into /I1/ or /I2 so that the final running disparity of the data code-group is negative. The first code group is /K28.5/ and the second code group is a data code-group other than /D21.5/ or /D2.2/. /I1/ (/K28.5/D5.6/) is used to flip the running disparity and /I2/ (/K28.5/D16.2/) is used to preserve the running disparity.

8B/10B Encoder Current Running Disparity Control Feature

The 8B/10B encoder performs a running disparity check on the 10-bit output data. The running disparity can also be controlled using `tx_forcedisp` and `tx_dispval`. When the PCS-PMA interface width is 10 bits, `tx_forcedisp` and `tx_dispval` are one bit each. When the PCS-PMA interface width is 20 bits, `tx_forcedisp` and `tx_dispval` are two bits each. The LSB of `tx_forcedisp` and `tx_dispval` corresponds to the LSByte of the input data and the MSB corresponds to the MSByte of the input data.

8B/10B Encoder Bit Reversal Feature

The bit reversal feature reverses the order of the bits of the input data. Bit reversal is performed at the output of the 8B/10B Encoder and is available even when the 8B/10B Encoder is disabled. For example, if the input data is 20-bits wide, bit reversal switches bit [0] with bit [19], bit [1] with bit [18] and so on.

8B/10B Encoder Byte Reversal Feature

The byte reversal feature is available only when the PCS-PMA interface width is 16 bits or 20 bits. Byte reversal is performed at the output of the 8B/10B Encoder and is available even when the 8B/10B Encoder is disabled. This feature swaps the LSByte with the MSByte and vice-versa. For example, when the PCS-PMA interface width is 16-bits, [7:0] bits (LSByte) gets swapped with [15:8] bits (MSByte) and [15:8] bits (MSByte) gets swapped with [7:0] bits (LSByte). As a result, the 16-bit bus becomes MSB to LSB, bits[7:0] to bits[15:8].

Polarity Inversion Feature

The polarity inversion feature is used in situations where the positive and the negative signals of a serial differential link are erroneously swapped during board layout. The polarity inversion feature inverts the value of each bit of the input data. For example, if the input data is 00101001, then the data gets changed to 11010110 after polarity inversion.

Pseudo-Random Binary Sequence (PRBS) Generator

The PRBS generator block generates PRBS patterns and square wave patterns.

PRBS test patterns may be considered equivalent to "noise." Use these pattern generators to test the transceiver link with a noisy signal (using the test patterns listed below) by placing the transceiver in loopback mode.

The PRBS generator supports a 64-bit PCS-PMA interface. PRBS9 supports 64-bit and 10-bit PCS-PMA interface widths to allow testing at lower data rates.

The following PRBS test patterns are supported:

- PRBS9: $x^9 + x^5 + 1$
- PRBS15: $x^{15} + x^{14} + 1$
- PRBS23: $x^{23} + x^{18} + 1$
- PRBS31: $x^{31} + x^{28} + 1$

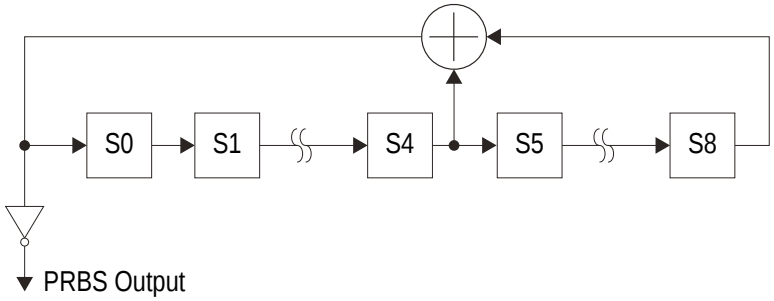
Enable the PRBS generator and select a test pattern through the reconfiguration interface.

Use PRBS9 to test transceiver links with linear impairments, and with 8B/10B.

Use PRBS15 for jitter evaluation.

Use PRBS23 or PRBS31 for jitter evaluation (data-dependent jitter) of non-8B/10B links, such as SDH/SONET/OTN jitter testers. Most 40G, 100G, and 10G applications use PRBS31 for link evaluation.

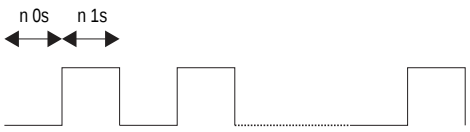
Figure 5-43: PRBS Generator for Serial Implementation of PRBS9 Pattern



Note: All supported PRBS generators are similar to the PRBS9 generator.

The square-wave generator supports 64-bit PCS-PMA interface widths. It has a programmable *n*-number of consecutive serial bit 1s and 0s, where *n* is 4, 6, or 8 (*n* defaults to 4).

Figure 5-44: Generator for Square Wave Pattern



Program the value of *n* through the Transceiver and PLL Address Map.

TX Bit Slip

The TX bit slip allows the word boundary to be controlled by `tx_std_bitslipboundarysel`. The TX bit slip feature is used in applications, such as CPRI, which has a data rate greater than 6 Gbps. The maximum number of the supported bit slips is PCS data width-1 and the slip direction is from MSB to LSB and from current to previous word.

Receiver Datapath

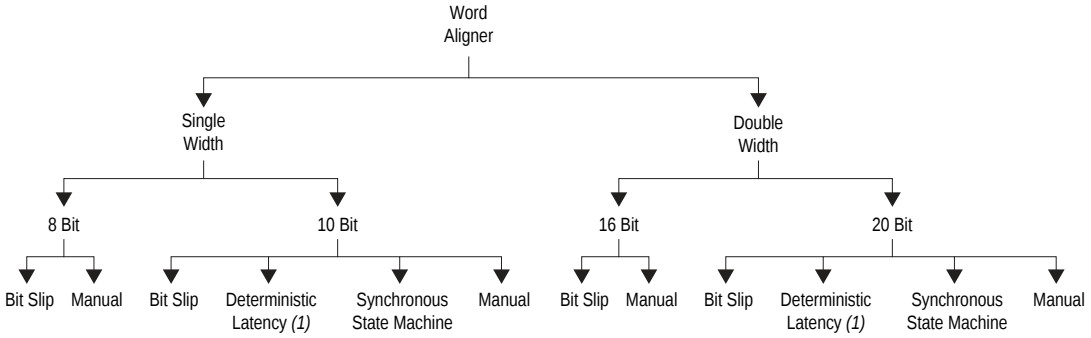
Word Aligner

The word aligner receives the serial data from the PMA and realigns the serial data to have the correct word boundary according to the word alignment pattern configured. This word alignment pattern can be 7, 8, 10, 16, 20, 32 and 40 bits in length.

Depending on your PCS-PMA interface width, the word aligner can be configured in one of the following modes:

- Bit slip
- Manual alignment
- Synchronous state machine
- Deterministic latency

Figure 5-45: Word Aligner Conditions and Modes



Note:
1. This option is available in CPRI mode.

Word Aligner Bit Slip Mode

In bit slip mode, the word aligner operation is controlled by `rx_bitslip`, which has to be held for two parallel clock cycles. At every rising edge of `rx_bitslip`, the bit slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. Pattern detection is not used in bit slipping mode; therefore, `rx_syncstatus` is not valid in this mode.

Word Aligner Manual Mode

In manual alignment mode, the word aligner operation is controlled by `rx_std_wa_patternalign`. The word aligner operation is edge-sensitive or level-sensitive to `rx_std_wa_patternalign`, depending upon the PCS-PMA interface width selected.

Table 5-6: Word Aligner `rx_std_wa_patternalign` Behavior

PCS-PMA Interface Width	`rx_std_wa_patternalign` Behavior
8	Rising edge sensitive
10	Level sensitive

PCS-PMA Interface Width	rx_std_wa_patternalign Behavior
16	Rising edge sensitive
20	Rising edge sensitive

If rx_std_wa_patternalign is asserted, the word aligner looks for the programmed word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If rx_std_wa_patternalign is deasserted, the word aligner maintains the current word boundary even when it sees the word alignment pattern in a new word boundary.

rx_syncstatus and rx_patterndetect, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate the word aligner status.

After receiving the first word alignment pattern after rx_std_wa_patternalign is asserted, both rx_syncstatus and rx_patterndetect are driven high for one parallel clock cycle. Any word alignment pattern received thereafter in the same word boundary causes only rx_patterndetect to go high for one clock cycle. Any word alignment pattern received thereafter in a different word boundary causes the word aligner to re-align to the new word boundary only if rx_std_wa_patternalign is asserted. The word aligner asserts rx_syncstatus for one parallel clock cycle whenever it re-aligns to the new word boundary.

Word Aligner Synchronous State Machine Mode

In synchronous state machine mode, when the programmed number of valid synchronization code groups or ordered sets is received, rx_syncstatus is driven high to indicate that synchronization is acquired. rx_syncstatus is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups, after which rx_syncstatus is driven low.

The word aligner indicates loss of synchronization (rx_syncstatus remains low) until the programmed number of valid synchronization code groups are received again.

Word Aligner Deterministic Latency Mode

The deterministic latency state machine in the word aligner reduces the known delay variation from the word alignment process and automatically synchronizes and aligns the word boundary by slipping a clock cycle in the deserializer. Incoming data to the word aligner is aligned to the boundary of the word alignment pattern (K28.5).

Using deterministic latency state machine mode and after the initial alignment following the deassertion of reset, rx_std_wa_patternalign must be reasserted to initiate another pattern alignment.

Table 5-7: PCS-PMA Interface Widths and Protocol Implementations

PCS-PMA Interface Width	Protocol Implementations
8	Basic
10	• Basic • Basic rate match • CPRI • PCIe Gen1 and Gen2 • GigE

PCS-PMA Interface Width	Protocol Implementations
16	Basic
20	• CPRI • Basic • Basic rate match

Word Aligner Pattern Length for Various Word Aligner Modes

Table 5-8: Word Aligner Pattern Length for Various Word Aligner Modes

PCS-PMA Interface Width	Supported Word Aligner Modes	Supported Word Aligner Pattern Lengths	`rx_std_wa_patternalign` behavior	`rx_syncstatus` behavior	`rx_patterndetect` behavior
8	Bit slip	7	N/A	N/A	N/A
	Manual	8, 16	Rising Edge Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new boundary	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary
10	Bit slip	7	N/A	N/A	N/A
	Manual	7, 10	Level Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new boundary	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary
	Deterministic latency (CPRI mode only)	10	—	—	—
	Synchronous State Machine	7, 10	N/A	Stays high as long as the synchronization conditions are satisfied.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary

PCS-PMA Interface Width	Supported Word Aligner Modes	Supported Word Aligner Pattern Lengths	rx_std_wa_patternalign behavior	rx_syncstatus behavior	rx_patterndetect behavior
16	Bit slip	16	N/A	N/A	N/A
	Manual	8, 16, 32	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on rx_std_wa_patternalign until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary
20	Bit slip	7	N/A	N/A	N/A
	Manual	7, 10, 20, 40	Rising edge sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on rx_std_wa_patternalign until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	Deterministic latency (CPRI mode only)	10	—	—	—
	Synchronous State Machine	7, 10, 20	—	Stays high as long as the synchronization conditions are satisfied	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Word Aligner RX Bit Reversal Feature

The RX bit reversal feature reverses the order of the data received from the PMA. It is performed at the output of the Word Aligner and is available even when the Word Aligner is disabled. If the data received from the PMA is a 10-bit data, the bit reversal feature switches bit [0] with bit [9], bit [1] with bit [8], and

so on. For example, if the 10-bit data is 1000010011, the bit reversal feature—when enabled—changes the data to 1100100001.

Word Aligner RX Byte Reversal Feature

The RX byte reversal feature is available only when the PCS-PMA interface width is 16 bits or 20 bits. This feature reverses the order of the data received from the PMA. RX byte reversal reverses the LSByte of the received data with its MSByte and vice versa. If the data received is 20-bits, bits[0..9] are swapped with bits[10..20] so that the resulting 20-bit data is [[10..20],[0..9]]. For example, if the 20-bit data is 11001100001000011111, the byte reversal feature changes the data to 10000111111100110000.

RX Polarity Inversion Feature

The RX polarity inversion feature inverts each bit of the data received from the PMA. If the data received is a 10-bit data. Bit[0] content is inverted to its complement, ~bit[0], bit[1] is inverted to its complement, ~bit[1], bit[2] is inverted to its complement, ~bit[2], and so on. For example, if the 10-bit data is 1111100000, the polarity inversion feature inverts it to 0000011111.

Rate Match FIFO

The rate match FIFO compensates for the frequency differences between the local clock and the recovered clock up to ± 300 ppm by inserting and deleting skip/idle characters in the data stream. The rate match FIFO has several different protocol specific modes of operation. All of the protocol specific modes depend upon the following parameters:

- Rate match deletion—occurs when the distance between the write and read pointers exceeds a certain value due to write clock having a higher frequency than the read clock.
- Rate match insertion—occurs when the distance between the write and the read pointers becomes less than a certain value due to the read clock having a higher frequency than the write clock.
- Rate match full—occurs when the write pointer wraps around and catches up to the slower-advancing read pointer.
- Rate match empty—occurs when the read pointer catches up to the slower-advancing write pointer.

Rate match FIFO operates in six modes:

- Basic single width
- Basic double width
- GigE
- PIPE
- PIPE 0 ppm
- PCIe

Related Information

- **How to Implement the Basic Rate Match Protocol Using the Arria 10 Transceiver Native PHY IP Core**
  For more information about implementing rate match FIFO for each mode.
- **Rate Match FIFO in Basic (Single Width) Mode** on page 2-238
  For more information about implementing rate match FIFO in basic single width mode.

- **Rate Match FIFO Basic (Double Width) Mode** on page 2-240
  For more information about implementing rate match FIFO in basic double width mode.
- **How to Implement GbE, GbE 1588 in Arria 10 Transceivers** on page 2-86
  For more information about implementing rate match FIFO in GigE mode.
- **PCI Express** on page 2-179
  For more information about implementing rate match FIFO in PCIe mode.
- **How to Implement PCI Express in Arria 10 Transceivers** on page 2-197
  For more information about implementing rate match FIFO in PIPE mode.

8B/10B Decoder

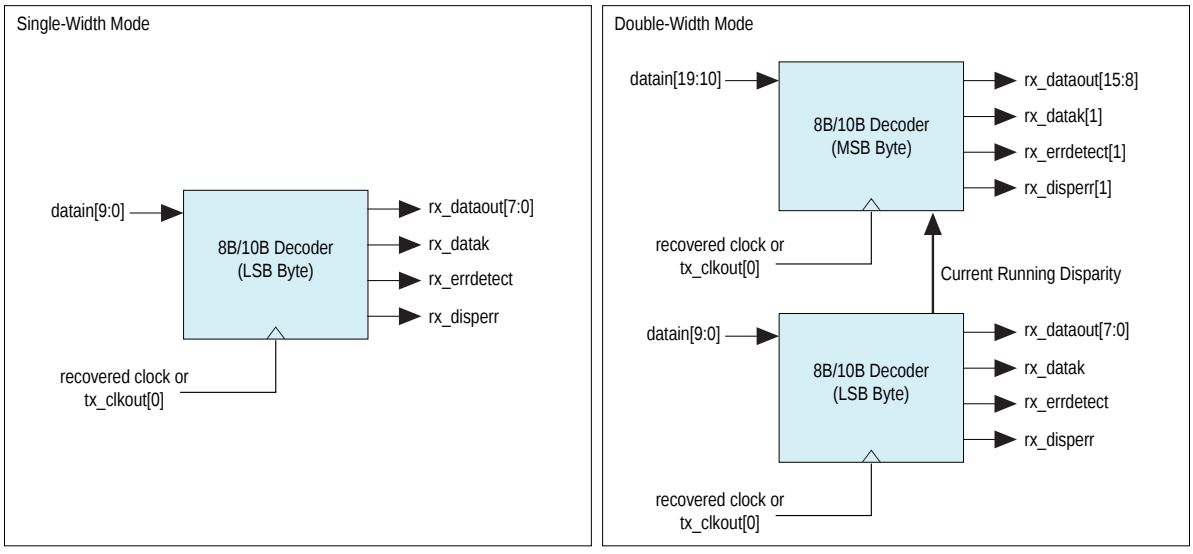
The general functionality for the 8B/10B decoder is to take a 10-bit encoded value as input and produce an 8-bit data value and a 1-bit control value as output. In configurations with the rate match FIFO enabled, the 8B/10B decoder receives data from the rate match FIFO. In configurations with the rate match FIFO disabled, the 8B/10B decoder receives data from the word aligner. The 8B/10B decoder operates in two conditions:

- When the PCS-PMA interface width is 10 bits and FGPA fabric-PCS interface width is 8 bits
- When the PCS-PMA interface width is 20 bits and FPGA fabric-PCS interface width is 16 bits

The 8B/10B decoder supports the following features:

- PCIe-PIPE polarity inversion
- Running disparity checker
- 8B/10B decoder bypass

Figure 5-46: 8B/10B Decoder in Single-Width and Double-Width Mode

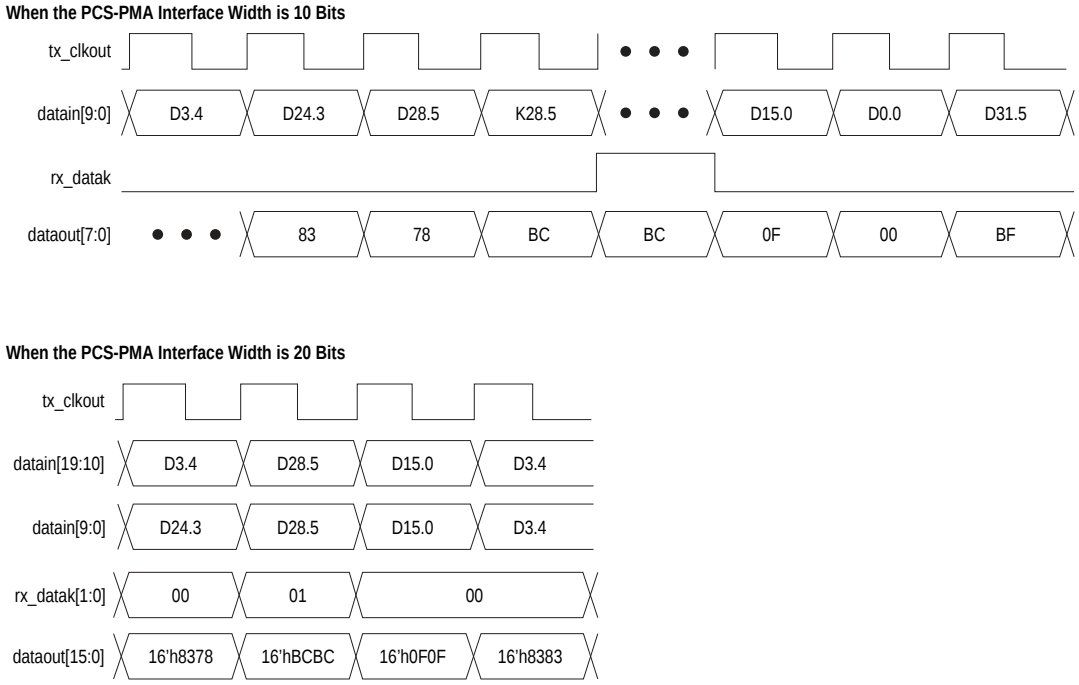


When the PCS-PMA interface width is 10 bits, only one 8B/10B decoder is used to perform the conversion. When the PCS-PMA interface width is 20 bits, two cascaded 8B/10B decoders are used. The 10-bit LSByte of the received 20-bit encoded data is decoded first and the ending running disparity is forwarded to the 8B/10B decoder responsible for decoding the 10-bit MSByte. The cascaded 8B/10B decoder decodes the 20-

bit encoded data into 16-bit data + 2-bit control identifier. The MSB and LSB of the 2-bit control identifier correspond to the MSByte and LSByte of the 16-bit decoded data code group. The decoded data is fed to the byte deserializer or the RX FIFO.

8B/10B Decoder Control Code Encoding

Figure 5-47: 8B/10B Decoder in Control Code Group Detection



The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on `rx_datak` . If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE 802.3 specification, `rx_datak` is driven high. If the received 10-bit code group is a data code group (/Dx.y/), `rx_datak` is driven low.

8B/10B Decoder Running Disparity Checker Feature

Running disparity checker resides in 8B/10B decoder module. This checker checks the current running disparity value and error based on the rate match output. `rx_runningdisp` and `rx_disperr` indicate positive or negative disparity and disparity errors, respectively.

Pseudo-Random Binary Sequence (PRBS) Verifier

The PRBS verifier checks the pseudo-random patterns generated by the PRBS generator. An error flag is asserted when an error occurs.

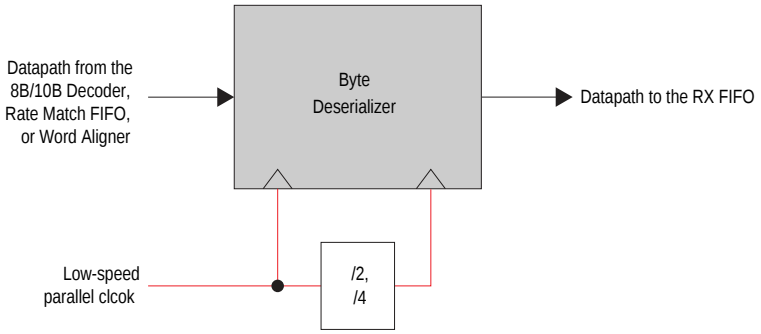
PRBS_DONE is asserted when the PRBS sequence has completed one full cycle. The PRBS verifier supports the following modes:

- PRBS 9 $G(x) = 1 + x^5 + x^9$
- PRBS 15 $G(x) = 1 + x^{14} + x^{15}$
- PRBS 23 $G(x) = 1 + x^{18} + x^{23}$
- PRBS 31 $G(x) = 1 + x^{28} + x^{31}$

Byte Deserializer

The byte deserializer allows the transceiver to operate at data rates higher than those supported by the FPGA fabric. It deserializes the recovered data by multiplying the data width two or four times, depending upon the deserialization mode selected. The byte deserializer is optional in designs that do not exceed the FPGA fabric interface frequency upper limit, and can be bypassed by using the **Disabled** option in the Quartus II Transceiver Native PHY. The byte deserializer operates in disabled, deserialize x2, or deserialize x4 mode.

Figure 5-48: Byte Deserializer Block Diagram



Byte Deserializer Disabled Mode

In disabled mode, the byte deserializer is bypassed. The data from the 8B/10B decoder, rate match FIFO, or word aligner is directly transmitted to the RX FIFO, depending on whether or not the 8B/10B decoder and rate match FIFO are enabled. Disabled mode is used in low speed applications such as GigE, where the FPGA fabric and the PCS can operate at the same clock rate.

Byte Deserializer Deserialize x2 Mode

The deserialize x2 mode is used in high-speed applications such as the PCIe Gen1 or Gen2 protocol implementation, where the FPGA fabric cannot operate as fast as the TX PCS.

In deserialize x2 mode, the byte deserializer deserializes 8-bit, 10-bit (when the 8B/10B encoder is not enabled), 16-bit, and 20-bit (when the 8B/10B encoder is not enabled) input data into 16-bit, 20-bit, 32-bit, and 40-bit data, respectively. As the parallel data width from the word aligner is doubled, the clock rate is halved.

Byte Deserializer Deserialize x4 Mode

The deserialize x4 mode is used in high-speed applications where the FPGA fabric cannot operate as fast as the TX PCS.

In deserialize x4 mode, the byte deserializer deserializes 8-bit data into 32-bit data. As the parallel data width from the word aligner is divided four times, the clock rate is quadrupled.

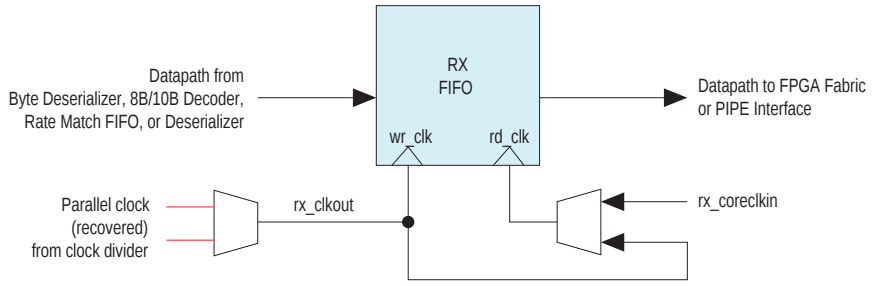
Bonded Byte Deserializer

The bonded byte deserializer is also available for channel-bundled applications such as PIPE. In this configuration, the control signals of the byte deserializers of all the channels are bonded together. A master channel controls all the other channels to prevent skew between the channels.

RX FIFO (Shared with Enhanced PCS and PCIe Gen3 PCS)

The RX FIFO interfaces between the PCS on the receiver side and the FPGA fabric and ensures reliable transfer of data and status signals. It compensates for the phase difference between the FPGA fabric and the PCS on the receiver side. It operates in register FIFO and low latency.

Figure 5-49: RX FIFO Block Diagram



RX FIFO Low Latency Mode

The low latency mode incurs two to three cycles of latency when connecting it with the FPGA fabric. The FIFO empty and the FIFO full threshold values are made closer so that the depth of the FIFO decreases, which in turn decreases the latency.

RX FIFO Register Mode

The register mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock. The register mode incurs only one clock cycle of latency when interfacing to the FPGA fabric.

Arria 10 PCI Express Gen3 PCS Architecture

Arria 10 architecture supports the PCIe Gen3 specification. Altera provides two options to implement the PCI Express solution:

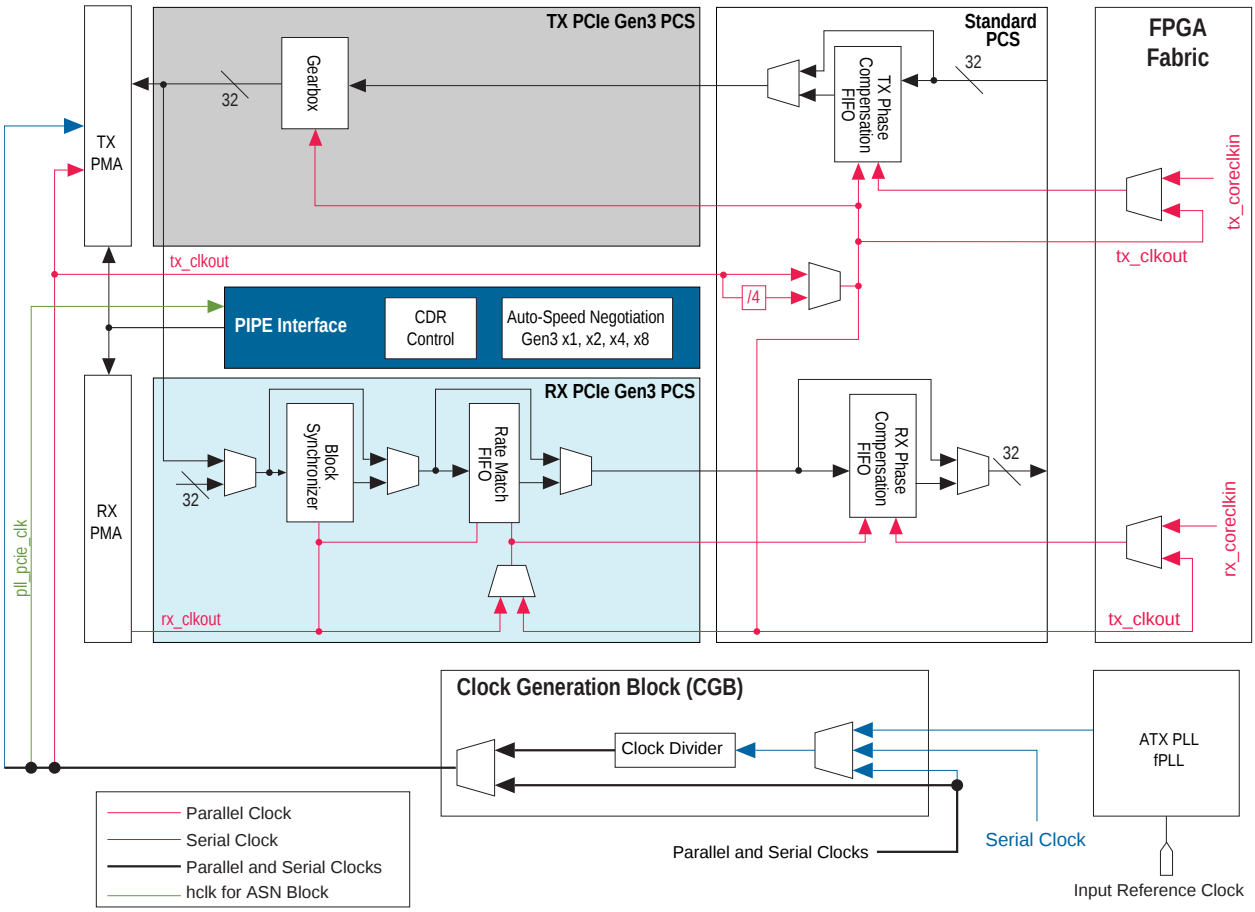
- You can use the Altera Hard IP solution. This complete package provides both the MAC Layer and the physical (PHY) layer functionality.
- You can implement the MAC in the FPGA core and connect this MAC to the transceiver PHY through the PIPE interface.

This section will focus on the basic blocks of PIPE 3.0-based Gen3 PCS architecture. The PIPE 3.0-based Gen3 PCS uses a 128b/130b block encoding/decoding scheme, which is different from the 8B/10B scheme

used in Gen1 and Gen2. The 130-bit block contains a 2-bit sync header and a 128-bit data payload. For this reason, Arria 10 devices include a separate Gen3 PCS that supports functionality at Gen3 speeds. This PIPE interface supports the seamless switching of Data and Clock between the Gen1, Gen2, and Gen3 data rates, and provides support for PIPE 3.0 features. The PCIe Gen3 PCS will support the PIPE interface with the Hard IP enabled, as well as with the Hard IP bypassed.

Note: For more information about Hard IP-based implementation, refer to the *Altera Hard IP for PCIe Users Guide*.

Figure 5-50: Gen3 PCS Block Diagram



Related Information

Altera Hard IP for PCIe Users Guide

PCI Express on page 2-179
For more information about PCIe Gen1, Gen2, and Gen3 implementation and configuration, refer to "Supported PIPE Features."

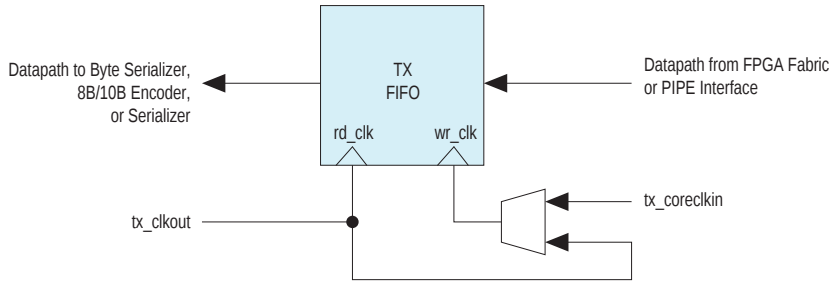
Transmitter Datapath

This section describes the TX FIFO and the Gearbox of the Gen3 PCS transmitter.

TX FIFO (Shared with Standard and Enhanced PCS)

The TX FIFO in each channel ensures a reliable transfer of data and status signals between the PCS channel and the FPGA fabric. The TX FIFO compensates for the phase difference between the low speed parallel PCS clock and the FPGA fabric clock. The RX and TX FIFOs are shared with standard and enhanced PCS. In Hard IP mode, the TX FIFO works in register mode. In PIPE mode, the TX FIFO works in low latency mode.

Figure 5-51: TX FIFO in Phase Compensation Mode



Concerning TX FIFO modes, PIPE Gen1, Gen2 and Gen3 only allow TX FIFO in Low Latency mode. The Low Latency mode incurs 3 - 4 cycles of latency when connecting with the FPGA Fabric. The FIFO empty and the FIFO full threshold values are made closer so that the depth of the FIFO decreases, which decreases the latency.

Related Information

Arria 10 Standard PCS Architecture on page 5-31
For more information about TX FIFO.

Gearbox

The PCIe 3.0 base specification specifies a block size of 130 bits, with the exception of the SKP Ordered Sets, which can be of variable length. An implementation of a 130-bit data path takes significant resources, so the PCIe Gen3 PCS data path is implemented as 32-bits wide. Because the TX PMA data width is fixed to 32 bits, and the block size is 130 bits with variations, a gearbox is needed to convert 130 bits to 32 bits.

The gearbox module in the TX PCS converts the 130-bit block to the 32-bit data required by the TX PMA as the data path implementation is 32 bits to reduce usage of resources. The 130-bit block is received as follows in the 32-bit data path: 34 (32 + 2-bit sync header), 32, 32, 32, 34 (32 + 2-bit sync header). During the first cycle the gearbox converts the 34-bit input data to 32-bit data. During the next 3 clock cycles the gearbox will merge bits from adjacent cycles to form the 32-bit data. In order for the gearbox to work correctly, a gap must be provided in the data for every 16 shifts as each shift is 2 bits for converting the initial 34-bit to 32-bit in the gearbox. After 16 Shifts the gearbox will have an extra 32-bit data that was transmitted out, and thus a gap is required in the input data stream. This gap is achieved by driving data valid low for one cycle after every 16 blocks of data.

Related Information

Gearbox on page 2-190

Receiver Datapath

This section describes the Block Synchronizer, Rate Match FIFO, and RX FIFO of the Gen3 PCS receiver.

Block Synchronizer

PMA parallelization occurs at arbitrary word boundaries. Consequently, the parallel data from the RX PMA CDR must be realigned to meaningful character boundaries. The PCI-Express 3.0 base specification outlines that the data is formed using 130-bit blocks, with the exception of SKP blocks.

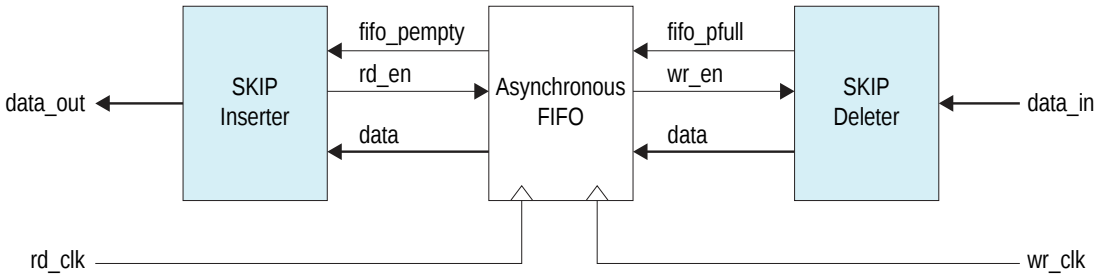
The SKP Ordered Set can be 66, 98, 130, 162, or 194 bits long. The block sync module searches for the Electrical Idle Exit Sequence Ordered Set (or the last number of fast training sequences (NFTS) Ordered Set) or skip (SKP) Ordered Set to identify the correct boundary for the incoming stream and to achieve the block alignment. The block is realigned to the new block boundary following the receipt of a SKP Ordered Set, as it can be of variable length.

Rate Match FIFO

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain to the local receiver reference clock domain. The rate match FIFO compensates for small clock frequency differences between these two clock domains by inserting or removing SKP symbols in the data stream to keep the FIFO from going empty or full respectively.

The PCI-Express 3.0 base specification defines that the SKP Ordered Set (OS) can be 66, 98, 130, 162, or 194 bits long. The SKP OS has the following fixed bits: 2-bit Sync, 8-bit SKP END, and a 24-bit LFSR = 34 Bits. The Rate Match/Clock compensation module adds or deletes the 4 SKP characters (32-bit) to keep the FIFO from going empty or full, respectively. This module monitors the `skip_found` signal from the block sync module, and if the FIFO is nearly full, deletes the 4 SKP characters (32-bit) by disabling write whenever a SKP is found. If the FIFO is nearly empty, the design waits for a SKP Ordered Set to start and then stops reading the data from the FIFO, and inserts a SKP in the outgoing data. The actual FIFO core (memory element) is in the Shared Memory block in the PCS channel.

Figure 5-52: Rate Match FIFO



RX FIFO (Shared with Standard and Enhanced PCS)

The RX FIFO in each channel ensures a reliable transfer of data and status signals between the PCS channel and the FPGA fabric. The RX FIFO compensates for the phase difference between the parallel PCS clock and the FPGA fabric clock. In PIPE mode, the RX FIFO works in low latency mode.

Related Information

Arria 10 Standard PCS Architecture on page 5-31
For more information about RX FIFO.

PIPE Interface

This section describes the Auto Speed Negotiation and the Clock Data Recovery Control of the Gen3 PIPE interface.

Auto Speed Negotiation

Auto speed negotiation controls the operating speed of the transceiver when operating under PIPE 3.0 modes. By monitoring the rate control signal from the PHY-MAC, this feature changes the transceiver from PIPE Gen1 operation mode to Gen2 operation mode, or from PIPE Gen1 operation mode to Gen2 operation mode to Gen3 operation mode, or vice versa. The switches among the Gen1, Gen2, and Gen3 data rates involve a reconfiguration of the PMA and PCS settings. The PMA must re-lock and provide a TX PLL clock, and its CDR will also lock at a new incoming data rate. The PIPE interface clock rate will also be adjusted to match the data throughput.

Related Information

Rate Switch on page 2-188

Clock Data Recovery Control

The CDR control feature is used for the L0s fast exit when operating in PIPE Gen3 mode. Upon detecting an Electrical Idle Ordered Set (EIOS), this feature takes manual control of the CDR by forcing it into a lock-to-reference mode. When an exit from electrical idle is detected, this feature moves the CDR into lock-to-data mode to achieve fast data lock.

Reconfiguration Interface and Dynamic Reconfiguration

6

2013.12.02

UG-A10XCVR

Subscribe Send Feedback

This chapter explains the purpose and the use of the Arria 10 reconfiguration interface that is part of the Transceiver Native PHY IP core and the Transceiver PLL IP core. You can use the reconfiguration interface to perform dynamic reconfiguration and to interface with blocks within the transceiver channel, such as the DFE and EyeQ blocks. Dynamic reconfiguration is the ability to dynamically modify transceiver channels and PLLs to meet changing requirements during device operation. Arria 10 transceiver channels and PLLs are fully customizable, allowing a system to adapt to its operating environment. You can customize channels and PLLs by dynamically triggering reconfiguration during device operation or following power-up.

Each transceiver channel and PLL contains an Avalon Memory-Mapped (Avalon-MM) reconfiguration interface. The reconfiguration interface provides direct access to the programmable space of each channel and PLL. Because each channel and PLL has its own dedicated Avalon interface, you can dynamically modify channels either concurrently or sequentially, depending upon how the Avalon master is connected to the Avalon interface. Communication with the channel and PLL reconfiguration interface requires an Avalon compliant master. This chapter provides complete information about using the reconfiguration interface.

Related Information

- **Interacting with the Reconfiguration Interface** on page 6-4
- **Reconfiguring Channel and PLL Blocks** on page 6-5
- **Switching Reference Clocks** on page 6-10
- **Transmitter PLL Switching** on page 6-9
- **Changing PMA Analog Parameters** on page 6-13
- **Using Data Pattern Generators and Checkers** on page 6-13

Ports and Parameters

The reconfiguration interface is integrated into the Native PHY instance and TX PLL. The following table lists the ports and parameters available for the Avalon Interface. You can define parameters for IP components by clicking **Tools** > **MegaWizard Plug-In Manager**. Dynamic reconfiguration is available for the Arria 10 Transceiver Native PHY, fPLL, ATX PLL, and the CMU PLL IP cores. To expose the ports of the reconfiguration interface, enable the **Enable reconfiguration** option when parameterizing the IP core.

Each transmit PLL and channel has a dedicated reconfiguration interface. The transmit PLL instance has a maximum of one reconfiguration interface. Unlike the PLL instance, the Native PHY instance can specify

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

multiple channels. For example, a two channel, Channel 0 and Channel 1, design would contain two reconfiguration interfaces. The reconfiguration interface ports are bundled together. The 32-bit wide ports are grouped into 32-bit groups. The 1-bit ports are grouped into 1-bit groups. For example, a two channel Native PHY IP instance address bus would be `reconfig_address[63:0]` and the reconfiguration clock would be `reconfig_clk[1:0]`. For the address bus, the channel 0 reconfiguration address would comprise of `reconfig_address[31:0]`. Channel 1 reconfiguration address would comprise of `reconfig_address[63:32]`. For the reconfiguration clock, channel 0's reconfiguration clock would be `reconfig_clk[0]` and channel 1 reconfiguration clock would be `reconfig_clk[1]`. The same bundling pattern applies to channel counts greater than two.

Table 6-1: Reconfiguration Interface Ports (with Share Reconfiguration Interface Disabled)

The following table lists the reconfiguration interface ports when **Shared reconfiguration interface** option is disabled. *N* equals the number of channels in the following table.

Port Name	Direction	Clock Domain	Description
`reconfig_clk[<N>-1:0]`	Input	N/A	Avalon clock. The preliminary clock frequency is 100 MHz to 125 MHz
`reconfig_reset[<N>-1:0]`	Input	`reconfig_clk`	Resets the Avalon interface.
`reconfig_write[<N>-1:0]`	Input	`reconfig_clk`	Write enable signal. Signal is active high.
`reconfig_read[<N>-1:0]`	Input	`reconfig_clk`	Read enable signal. Signal is active high.
`reconfig_address[10*<N>-1:0]`	Input	`reconfig_clk`	The lower 10 bits are the address lines. The upper bits specify the channel.
`reconfig_writedata[32*<N>-1:0]`	Input	`reconfig_clk`	A 32-bit data write bus. Data to be written into the address indicated by `reconfig_address`.
`reconfig_readdata[32*<N>-1:0]`	Output	`reconfig_clk`	A 32-bit data read bus. Valid data is placed on this bus after a read operation. Signal is valid after `waitrequest` goes high and then low.
`reconfig_waitrequest[<N>-1:0]`	Output	`reconfig_clk`	A one bit signal that indicates the Avalon interface is busy. Issue no Avalon commands while this signal is high.

If you enable **Share reconfiguration interface** for a 2 channel instance, the `reconfig_address` bus becomes an 11-bit bus `reconfig_address[10:0]`. The lower 10 bits, `[9:0]`, is the reconfiguration address. The uppermost bit, `[10]`, indicates the active channel. A value of 0 indicates channel 0 and a value of 1 indicates channel 1.

Table 6-2: Reconfiguration Interface Ports (with Share Reconfiguration Interface Enabled)

The following table lists the reconfiguration interface ports when **Shared reconfiguration interface** option is enabled. *N* equals the number of channels in the following table.

Port Name	Direction	Clock Domain	Description
`reconfig_clk[0:0]`	Input	N/A	Avalon clock.

Port Name	Direction	Clock Domain	Description
reconfig_reset[0:0]	Input	reconfig_clk	Resets the Avalon interface.
reconfig_write[0:0]	Input	reconfig_clk	Write enable signal. Signal is active high.
reconfig_read[0:0]	Input	reconfig_clk	Read enable signal. Signal is active high.
reconfig_address[log2<*N*>+9:0]	Input	reconfig_clk	A 10-bit address bus. Used in read and write operations.
reconfig_writedata[31:0]	Input	reconfig_clk	A 32-bit data write bus. Data to be written into the address indicated by reconfig_address.
reconfig_readdata[31:0]	Output	reconfig_clk	A 32-bit data read bus. Valid data is placed on this bus after a read operation. Signal is valid after waitrequest goes high and then low.
reconfig_waitrequest[0:0]	Output	reconfig_clk	A one bit signal that indicates the Avalon interface is busy. Issue no Avalon commands while this signal is high.

Table 6-3: Avalon Interface Parameters

Specify values for the following parameters in the **Dynamic Reconfiguration** tab of the Transceiver Native PHY and TX PLL parameter GUIs.

Parameter	Value	Description
Enable dynamic reconfiguration	Enable / Disable	Enables the reconfiguration interface. Disabled by default. The reconfiguration interface is exposed when this option is enabled.
Share reconfiguration interface	Enable / Disable	Use a single reconfiguration interface to control all channels. Disabled by default. If enabled, the upper most bits of the reconfig_address identifies the active channel. The lower 10-bits specify the reconfiguration address. Binary encoding is used to identify the active channel (available only for Transceiver Native PHY)
Enable embedded JTAG AVMM master	Enable / Disable	Includes a JTAG Avalon-MM master to control the reconfiguration interface. For example, use System Console to control the JTAG Avalon-MM master. Disabled by default.
Generate SystemVerilog package file	Enable / Disable	Creates a SystemVerilog package file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.
Generate C header file	Enable / Disable	Creates a C header file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.
Generate MIF (Memory Initialize File)	Enable / Disable	Creates a MIF file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.

Interacting with the Reconfiguration Interface

This section describes how to interact with the reconfiguration interface by performing Avalon read and write operations to initiate dynamic reconfiguration of specific transceiver parameters. All read and write operations must adhere to the Avalon specification.

Related Information

- *Avalon Interface Specification*

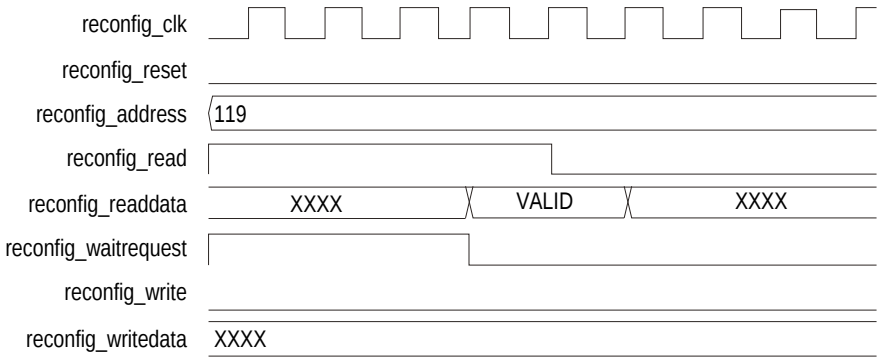
Performing a Read to the Reconfiguration Interface

Reading to the reconfiguration interface prompts the Avalon master to determine the current value at a specific address. The read operation involves the following steps:

1. Place a 10-bit feature address on the `reconfig_address` bus.
2. Assert the `reconfig_read` signal.

After the `reconfig_read` signal asserts, the `reconfig_waitrequest` signal always asserts for a number of `reconfig_clock` cycles, then deasserts. This deassertion indicates the `reconfig_readdata` bus contains valid data.

Figure 6-1: Read Operation



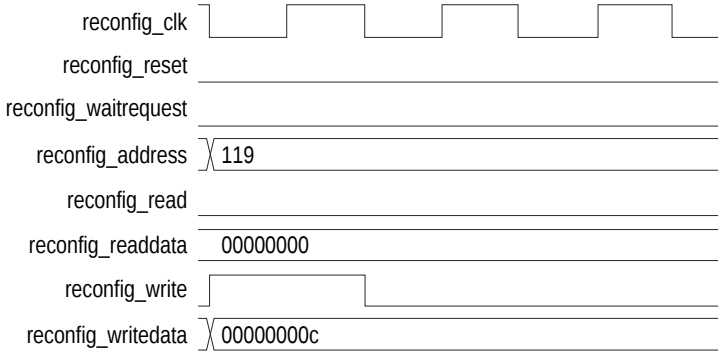
Performing a Write to the Reconfiguration Interface

Writing to the reconfiguration interface of either the Transceiver Native PHY or TX PLL allows an Avalon master to change the value at a specific address. All writes performed with the reconfiguration interface must be a read-modified-write, because two or more features may share the same reconfiguration address. When two or more features share the same reconfiguration address, one feature's data bits are interleaved with another feature's data bits. Therefore, you must always use read-modified-write and the correct address and data bits.

Read-modify-write operation involves the following steps:

1. Place a 10-bit feature address on the `reconfig_address` bus.
2. Assert the `reconfig_read` signal and storing the read value. Wait for `reconfig_waitrequest` to transition from 1 to 0 before reading the value on the `reconfig_readdata` bus.
3. From the read value, modify only the necessary bits by masking out the non-relevant bits. Store this new value internally using the additional steps below. Wait for the high to low transition on the waitrequest signal to indicate valid data is on the readdata bus.
4. Place the same 10-bit feature address from step 1 on the `reconfig_address` bus and place the new 32-bit value on the `reconfig_writedata` bus.
5. Assert the `reconfig_write` signal.
6. Reset the transceiver channel as appropriate.

Figure 6-2: Write Operation



Reconfiguring Channel and PLL Blocks

Certain systems may require that your transceiver channels or PLLs take on a different data rate or protocol specification. For example, the 8b/10b encoder may be enabled under one mode of operation, and disabled in another mode of operation. The following other examples also require channel and PLL block reconfiguration:

- Switching between the Standard and Enhanced PCS
- Changing the transmit PLL frequency
- Enabling and disabling blocks within a PCS block

You can dynamically reconfigure blocks within the transceiver channel or PLL through the reconfiguration interface. Reconfiguration of the channel and PLL blocks requires the following steps:

1. Generate required configuration files
2. Determine address offsets and differences
3. Perform read-modify-writes
4. Reset transceiver channels and PLLs as appropriate

Step 1: Generate Required Configuration Files

Dynamic reconfiguration requires two instances of the Altera Transceiver Native PHY IP or PLL: one instance defines the base transceiver or PLL configuration, and the second instance defines the modified configuration.

Click **Tools** > **MegaWizard Plug-In Manager** to define base and modified instances of the Transceiver Native PHY or PLL IP, according to the following:

Table 6-4: Transceiver Native PHY or PLL IP Parameters (Base and Modified Instances)

Native PHY Instance	Required Parameter Settings	Saved In
Base Configuration	• Click **Interfaces** > **Transceiver PHY** > **Arria 10 Transceiver Native PHY** for the Native PHY. Or, select one of the supported transmit PLL IP cores under **PLL**. Enable all options required for the base configuration, such as data rate, PCS options, and PMA options. • Enable all ports to be used by the modified configuration. For example, if the bitslip feature is not required in the base configuration, but required in modified configuration, then you must enable the `tx_std_bislipboundarysel` port. Reconfiguring between Standard PCS and Enhanced PCS requires **Enable reconfiguration between Standard and Enhanced PCS datapaths** be enabled. The **Transceiver configuration rules** define the initial mode of the PHY instance. • On the **Dynamic Reconfiguration** tab, turn on **Enable dynamic reconfiguration** and specify the **Configuration Options**. This flow uses the **Generate SystemVerilog package file** option.	• *<Native PHY Base Instance Name>***/reconfig/altera_xcvr_native_a10_reconfig_parameters.sv** contains all transceiver addresses and their bit value for that transceiver configuration. Or • *<PLL Base Instance Name>***/reconfig/altera_xcvr_<type>_pll_a10_reconfig_parameters.sv** contains all PLL addresses and their bit value for that PLL configuration.
Modified Configuration	• Click **Interfaces** > **Transceiver PHY** > **Arria 10 Transceiver Native PHY**. Or, select one of the supported transmit PLL IP cores under **PLL**. Enable all options required for the base configuration, such as data rate, PCS options, and PMA options. Reconfiguring between Standard PCS and Enhanced PCS requires **Enable reconfiguration between Standard and Enhanced PCS datapaths** be enabled. The **Transceiver configuration rules** define the initial mode of the PHY instance. • Enable all ports to be used by the modified configuration. • On the **Dynamic Reconfiguration** tab, turn on **Enable dynamic reconfiguration** and specify the same **Configuration Options** as the base instance.	• *<Native PHY Modified Instance Name>***/reconfig/altera_xcvr_native_a10_reconfig_parameters.sv** contains all transceiver addresses and their bit value for that transceiver configuration. Or • *<PLL Modified Instance Name>***/reconfig/altera_xcvr_<type>_pll_a10_reconfig_parameters.sv** contains all PLL addresses and their bit value for that PLL configuration.

Note: You can generate the base and modified instance files in the same or different folders. If you use the same folder then each configuration instance name must be unique.

Related Information

Using Configuration Files on page 6-7

Step 2: Determine Address Offsets and Differences

Use a text editor to compare the differences between the base and modified Transceiver Native PHY or PLL instance files. The differences between these two files indicate the addresses offset and bit values that must change to switch from one configuration to another. Some features span multiple addresses in the configuration file. Therefore, enabling or disabling a single feature can result in multiple address reads and writes to the reconfiguration interface.

Step 3: Perform Read-Modify-Writes

After programming the device with the base configuration, you can reconfigure the transceiver with the modified configuration. You must issue a read-modify-write to the reconfiguration interface for each address and bit difference between the base and modified Transceiver Native PHY or PLL. Write both the address and bit differences to the reconfiguration interface.

Step 4: Reset Transceiver Channels

After performing all read-modify-write operations, you must implement the required reset sequence. Refer to the required reset sequence to determine which blocks require reset.

Using Configuration Files

You can save the parameters you specify for the Transceiver Native PHY and transmit PLL IP instances as configuration files.

The configuration files store offsets along with the data values required for a particular transceiver instance. You can use the configuration files as part of dynamic reconfiguration flow.

Specify the file type of the configuration file on the **Dynamic Reconfiguration** tab of either the Transceiver Native PHY or transmit PLL parameter GUI. Select one or more of **System Verilog package file**, **C header file**, or **MIF** file types. All configuration files are stored in **<IP instance name>/reconfig/**. All configuration files generated for a particular IP instance contain the same configuration data.

The configuration file generates in the format you select. For example, the System Verilog package file type generates a System Verilog (**.sv**) file type that contains a two dimension data array. The data array stores either channel or PLL offsets, depending on the configuration file generated, including bitmasks and bit values.

Typical SystemVerilog Configuration File Line

```
    26'h008FF04,
// [25:16]-DPRIO address=0x008;
// [15:8]-bit mask=0xFF;
// [7:7]-
hssi_tx_pcs_pma_interface_pldif_datawidth_mode=pldif_data_10bit(1'h0);

// [6:5]-hssi_tx_pcs_pma_interface_tx_pma_data_sel=ten_g_pcs(2'h0);
// [4:4]-hssi_tx_pcs_pma_interface_prbs_gen_pat=prbs_gen_dis(1'h0);
// [3:0]-hssi_tx_pcs_pma_interface_sq_wave_num=sq_wave_default(4'h4);
```

...

```
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_VALUE =
"pldif_data_10bit";
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_OFST =
 8;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_OFST = 7;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_HIGH = 7;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_SIZE = 1;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_BITMASK =
 32'h00000080;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_VALMASK =
 32'h00000000;
localparam
HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_VALUE = 1'h0;
```

The System Verilog configuration files are separated into two parts that represent the same data using two formats. The first part of the configuration file consists of a data array of 26-bit hexadecimal values, and the second part consists of parameter values. For the data array, each 26-bit hexadecimal value contains an associated comment describing the various bit positions.

Table 6-5: 26-bit Value of Typical SystemVerilog Configuration File Line

Bit Position	Description
[25:16]	The channel or PLL offset.
[15:8]	The channel or PLL bit mask. The bit mask exposes the bits that are configured by either the Transceiver Native PHY or the transmit PLL IP cores.
[7:0]	Feature bit values. Multiple features may reside at the same offset or span multiple offsets.

Using the value of 26'h008FF04, the offset value is 0x008 and the bit mask is 0xFF. The four features that reside at offset 0x008 are:

- `hssi_tx_pcs_pma_interface_pldif_datawidth_mode` with a value of 1'h0
- `hssi_tx_pcs_pma_interface_tx_pma_data_sel` with a value of 2'h0
- `hssi_tx_pcs_pma_interface_prbs_gen_pat` with a value of 1'h0
- `hssi_tx_pcs_pma_interface_sq_wave_num` with a value of 4'h4

Writing to bit[7] of offset 0x008 changes the `hssi_tx_pcs_pma_interface_pldif_datawidth_mode` feature. The same data array information is also represented using the System Verilog parameter keyword, `localparam`. The C header file is setup similar to the System Verilog package file. The consist of an unsigned data array of 26-bit hexadecimal values and the second part consists of constant values. Both the data array and the constant values represent the same values. The MIF is setup is setup similar to the data array in the System Verilog package file and the C header file.

Altera recommends following the flow described in section *Reconfiguring Transceiver and PLL Blocks* when performing dynamic reconfiguration of either the Native PHY or transmit PLL.

Related Information

Reconfiguring Channel and PLL Blocks on page 6-5

Transmitter PLL Switching

Dynamically switching data rates increases system flexibility to support multiple protocols. You can change the transceiver channel data rate by switching the PLL. You can clock transceiver channels with up to four different transmitter PLLs. You can use the reconfiguration interface to specify which PLL drives the transceiver channel. The PLL switching method remains the same, regardless of the number of transmitter PLLs involved.

Before initiating the PLL switch procedure, ensure that your Transceiver Native PHY instance defines more than one transmitter PLL input. Specify the **Number of TX PLL clock inputs per channel** parameter on the **TX PMA** tab during Transceiver Native PHY parameterization.

The following table shows mapping of the Native PHY serial clock inputs to their respective logical mapping. The reconfiguration process requires this logical mapping. The number of exposed `tx_serial_clk` varies according to the number of transmitter PLLs you specify. The following table shows the logical mapping and the selection MUX offsets. Use the Native PHY reconfiguration interface for this operation.

Table 6-6: Logical Mapping of Native PHY Serial Clock Inputs

Transceiver Native PHY Port	Description	Logical PLL Offset	4-bit Logical PLL Offset Bits
`tx_serial_clk0`	Represents logical `PLL0`	0x117	[3:0]
`tx_serial_clk1`	Represents logical `PLL1`	0x117	[7:4]
`tx_serial_clk2`	Represents logical `PLL2`	0x118	[3:0]
`tx_serial_clk3`	Represents logical `PLL3`	0x118	[7:4]
N/A	PLL selection MUX	0x111	[7:0]

When performing a PLL switch, you must specify the replacement logical PLL offset and respective and bits. The following procedure describes selection of a specific transmitter PLL when more than one PLL is connected to a channel. To change the data rate of the CDR, follow the detailed steps for reconfiguring channel and PLL blocks. After determining the logical PLL to switch to, follow this procedure to switch to the selected transmitter PLL

1. Read from the selected logical PLL offset (for example, offset 0x117 or 0x118) and save the required 4-bit pattern. For example, switching to logical `PLL1` requires saving bits [7:4] of offset 0x117.
2. Encode the 4-bit value read in the previous step into an 8-bit value according to the following table:

Table 6-7: Logical PLL Encoding

4-bit Logical PLL Offset Bits	8-bit Mapping to Offset 0x111
[3..0]	{*~logical_PLL_offset_readdata[3]*, *logical_PLL_offset_readdata[1:0]*,*logical_PLL_offset_readdata[3]*, *logical_PLL_offset_readdata[3:0]* }

4-bit Logical PLL Offset Bits	8-bit Mapping to Offset 0x111
[7..4]	{*~logical_PLL_offset_readdata[7]*, *logical_PLL_offset_readdata[5:4]*,*logical_PLL_offset_readdata[7]*, *logical_PLL_offset_readdata[7:4]* }

Note: For example, if reconfiguring to logical `PLL1` then bits [7:4] is encoded to an 8-bit value {~bit[7], bit[5:4], bit[7], bit[7:4]}.

3. Perform a read-modify-write to bits[7:0] of offset 0x111 using the encoded 8-bit value.
4. Complete the required reset sequence.

Related Information

Reconfiguring Channel and PLL Blocks on page 6-5

Switching Reference Clocks

You can dynamically switch the input clock source to the ATX PLL, the fPLL, the CMU, and the CDR.

ATX Reference Clock Switching

You can use the reconfiguration interface on the ATX PLL instance to specify which reference clock source drives the ATX PLL. The ATX PLL supports clocking by up to five different reference clock sources. Independent of the number of transmitter PLLs specified, the flow to select between the different reference clock sources remain the same.

Before initiating a reference clock switch, ensure that your ATX PLL instance defines more than one reference clock source. Specify the **Number of PLL reference clocks** parameter on the **PLL** tab during ATX PLL parameterization.

The following table shows mapping of the ATX PLL reference clock inputs to their respective logical mapping. The reconfiguration process requires this logical mapping. The number of exposed `pll_refclk` ports varies according to the number of reference clocks you specify. The following table shows the logical mapping and the selection MUX offsets. Use the ATX PLL reconfiguration interface for this operation.

Table 6-8: Logical Mapping of ATX PLL Reference Clock Inputs

Transceiver ATX PLL Port	Description	Logical Reference Clock Offset	Bits
`pll_refclk0`	Represents logical `refclk0`	0x113	[7:0]
`pll_refclk1`	Represents logical `refclk1`	0x114	[7:0]
`pll_refclk2`	Represents logical `refclk2`	0x115	[7:0]
`pll_refclk3`	Represents logical `refclk3`	0x116	[7:0]
`pll_refclk4`	Represents logical `refclk4`	0x117	[7:0]
N/A	ATX Refclk selection MUX	0x112	[7:0]

When performing a reference clock switch, you must specify the logical reference clock and respective offset and bits of the replacement clock. After determining the ATX PLL, follow this procedure to switch to the selected reference clock:

1. Read from the logical reference clock offset and save the required 8-bit pattern. For example, switching to logical `refclk2` requires use of bits `[7:0]` of offset `0x115`.
2. Perform a read-modify-write to bits `[7:0]` of offset `0x112` using the 8-bit value obtained from the logical `refclk` offset.
3. Complete the required reset sequence.

fPLL Reference Clock Switching

You can use the reconfiguration interface on the fPLL instance to specify which reference clock source drives the fPLL. The fPLL supports clocking by up to five different reference clock sources. Independent of the number of transmitter PLLs specified, the flow to select between the different reference clock sources remain the same.

Before initiating a reference clock switch, ensure that your f PLL instance defines more than one reference clock source. Specify the **Number of PLL reference clocks** parameter on the **PLL** tab during fPLL parameterization.

The following table shows mapping of the fPLL reference clock inputs to their respective logical mapping. The reconfiguration process requires this logical mapping. The number of exposed `pll_refclk` ports varies according to the number of reference clocks you specify. The following table shows the logical mapping and the selection MUX offsets. Use the fPLL reconfiguration interface for this operation.

Table 6-9: Logical Mapping of fPLL Reference Clock Inputs

Transceiver ATX PLL Port	Description	Logical Reference Clock Offset	Bits
`pll_refclk0`	Represents logical `refclk0` for MUX_0	0x117	[4:0]
`pll_refclk1`	Represents logical `refclk1` for MUX_0	0x118	[4:0]
`pll_refclk2`	Represents logical `refclk2` for MUX_0	0x119	[4:0]
`pll_refclk3`	Represents logical `refclk3` for `MUX_0`	0x11A	[4:0]
`pll_refclk4`	Represents logical `refclk4` for `MUX_0`	0x11B	[4:0]
N/A	fPLL Refclk selection `MUX_0`	0x114	[4:0]
`pll_refclk0`	Represents logical `refclk0` for `MUX_1`	0x11D	[4:0]
`pll_refclk1`	Represents logical `refclk1` for `MUX_1`	0x11E	[4:0]
`pll_refclk2`	Represents logical `refclk2` for `MUX_1`	0x11F	[4:0]
`pll_refclk3`	Represents logical `refclk3` for `MUX_1`	0x120	[4:0]
`pll_refclk4`	Represents logical `refclk4` for `MUX_1`	0x121	[4:0]
N/A	fPLL Refclk selection `MUX_1`	0x11C	[4:0]

Specify the logical reference clock and respective offset and bits of the replacement clock when performing a reference clock switch. Follow this procedure to switch to the selected reference clock:

1. Read from the logical reference clock offset for MUX A and save the required 5-bit pattern. For example, switching to logical `refclk3` requires use of bits[4:0] of offset 0x11A.
2. Perform a read-modify-write to bits [4:0] of offset 0x114 using the 5-bit value obtained from the logical refclk offset.
3. Read from the logical reference clock offset for MUX B and save the required 5-bit pattern. For example, switching to logical `refclk3` requires use of bits[4:0] of offset 0x120.
4. Perform a read-modify-write to bits [4:0] of offset 0x11C using the 5-bit value obtained from the logical refclk offset.
5. Complete the required reset sequence.

CDR and CMU Reference Clock Switching

You can use the reconfiguration interface to specify which reference clock source drives the CDR and CMU PLL. The CDR supports clocking by up to five different reference clock sources.

Before initiating a reference clock switch, ensure that your Transceiver Native PHY or CMU PLL instance defines more than one reference clock source. Specify the **Number of CDR reference clocks** under the **RX PMA** tab when parameterizing the Native PHY.

Before initiating a reference clock switch, ensure that your CDR and CMU defines more than one reference clock source. For the CDR, specify the parameter on the **PLL** tab during ATX PLL parameterization. For the CMU, specify the **Number of PLL reference clocks** under the **PLL** tab when parameterizing the CMU PLL.

The following table describes the mapping of the CDR reference clock inputs to their respective logical mapping. The reconfiguration process requires the logical mapping. The number of exposed `rx_cdr_refclk` (CDR) or `pll_refclk` (CMU) varies according to the number of reference clocks you specify. The following table shows the logical mapping and the selection MUX offsets. Use the Native PHY reconfiguration interface for switching the CDR reference clock. Use the CMU reconfiguration interface for switching the CMU reference clock.

Table 6-10: Logical Mapping of CDR Reference Clock Inputs

Native PHY Port	Description	Logical Reference Clock Offset	Bits
`cdr_refclk0`	Represents logical `refclk0`	0x16A	[7:0]
`cdr_refclk1`	Represents logical `refclk1`	0x16B	[7:0]
`cdr_refclk2`	Represents logical `refclk2`	0x16C	[7:0]
`cdr_refclk3`	Represents logical `refclk3`	0x16D	[7:0]
`cdr_refclk4`	Represents logical `refclk4`	0x16E	[7:0]
N/A	CDR Refclk selection MUX	0x141	[7:0]

When performing a reference clock switch, note the logical reference clock to switch to and the respective offset and bits. After determining the logical reference clock, follow this procedure to switch to the selected CDR reference clock:

1. Read from the logical reference clock offset and save the required 8-bit pattern. For example, switching to logical `refclk3` requires saving bits `[7:0]` of offset `0x16D`.
2. Perform a read-modify-write to bits `[7:0]` of offset `0x141` using the 8-bit value obtained from the logical `refclk` offset.
3. Complete the required reset sequence.

Changing PMA Analog Parameters

You can use the reconfiguration interface on the Transceiver Native PHY IP to change the value of PMA analog features. Transceiver Native PHY IP configuration files do not contain analog settings. Set analog parameters using Quartus II Settings File (**.qsf**) variables or these steps.

To change any of the PMA analog features:

1. Read from the PMA analog feature offset of the channel you want to change. For example, change pre-emphasis 1st post-tap read and store the value of offset 0x105.
2. Select a valid value for the feature according to the Arria 10 register map. For example, a valid setting for pre-emphasis 1st post-tap is fir_post_1t_l1 which has a bit encoding of 6'b000001.
3. Perform a read-modify-write to the offset of the PMA analog feature using the valid value. For example, changing the pre-emphasis 1st post-tap to setting fir_post_1t_l1 you must write 6'b000001 to offset 0x105.

Table 6-11: PMA Analog Feature Offsets

PMA Analog Feature	Offset	Bit
Pre-emphasis 1 st post-tap	0x105	[5:0]
Pre-emphasis 1 st post-tap polarity	0x105	[6]
Pre-emphasis 2 st post-tap	0x106	[4:0]
Pre-emphasis 2 st post-tap polarity	0x106	[5]
Pre-emphasis 1 st pre-tap	0x107	[4:0]
Pre-emphasis 1 st pre-tap polarity	0x107	[5]
Pre-emphasis 2 st pre-tap	0x108	[3:0]
Pre-emphasis 2 st pre-tap polarity	0x108	[4]
Differential output voltage (Vod)	0x109	[5:0]
Pre-emphasis 2 st pre-tap polarity	0x109	[6]

Using Data Pattern Generators and Checkers

The Arria 10 transceivers contain hardened data generators and checkers to provide a simple and easy way to verify and characterize high speed links. Hardening the data generators and verifiers saves FPGA core logic resources. The pattern generator block supports the following patterns:

- Pseudo Random Binary Sequence (PRBS)
- Pseudo Random Pattern (PRP)
- Square wave

Note: The pattern generators and verifiers are supported only for non-bonded channels.

Using PRBS and Square Wave Data Pattern Generator and Checker

You can use PRBS to simulate traffic and easily characterize high-speed links without developing or fully implementing any upper layer of a protocol stack.

The PRBS generator generates a self-aligning pattern that doesn't require bit slipping or word alignment for a pattern lock. Because the PRBS pattern is generated by an LFSR, the next pattern can be determined based upon the previous pattern. This is important for the PRBS checker because a portion of the received pattern can be used to generate the next sequence of bits to verify the next data sequence received is correct.

The PRBS generator and checker support two data width configurations: 64-bit and 10-bit. The PRBS generator, PRBS checker, and square wave must be configured to a supported bus width (that is, 64-bit or 10-bit, independent of the FPGA-PCS fabric width used). For example, if your FPGA-PCS fabric width is set to a 40-bit width, then you must configure the PRBS or square wave to a valid 64-bit or 10-bit width data generator.

Table 6-12: PRBS and Square Wave Supported Polynomials and Data Widths

Pattern	Polynomial	64-Bit	10-Bit
PRBS 9	G(x) = 1+ x5 + x9	X	x
PRBS 15	G(x) = 1+ x14 + x15	X	
PRBS 23	G(x) = 1+ x18 + x23	x	
PRBS 31	G(x) = 1+ x28 + x31	x	
Square Wave	Number of consecutive 1's and 0's: 4,6,8	x	

Enabling the PRBS and Square Wave Data Generator

You must perform a sequence of read-modify-writes to the Native PHY reconfiguration interface to enable either the PRBS or square wave data generator. You must perform the read-modify-writes to offsets: 0x6, 0x7, 0x8, and 0x110. To enable either the PRBS or the square wave data generator, follow these steps:

1. Perform a read-modify-write to offset 0x6 according to the **Table 6-13**.
2. Perform a read-modify-write to either offset 0x7 or offset 0x8 to enable the type of PRBS or square wave pattern:
   a. Perform a read-modify-write to offset 0x7 by writing a 1'b1 to either bit[7], bit[6], or bit[5] to enable PRBS9, PRBS15, or PRBS23, respectively. Write other bits with 1'b0.
   b. Perform a read-modify-write to offset 0x8 by writing to bit[3:0], or bit[4] to enable the square wave or PRBS31, respectively. Write bit[6:5] with the type pattern generator.
3. Perform a read-modify-write to offset 0x110 with the specified width. This data width is either 64-bit or 10-bit.
4. Perform a reset.
   To disable the PRBS or square wave generator, write the original values back into the read-modify-write offsets in **Table 6-12**.

Table 6-13: PRBS Generator and Square Wave Offsets

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Other Offsets	Attribute Encoding	Bit Encoding	Description
0x6	[2:0]	tx_pma_data_sel	0x8	prbs_pat	3'b100	Enable PRBS generator
				sq_wave_pat	3'b101	Enable square wave generator
	[3]	prbs9_dwidth		prbs9_10b	1'b1	Enable PRBS9 in 10-bit mode
				prbs9_64b	1'b0	Enable PRBS9 in 64-bit mode
	[6]	prbs_clken		prbs_clk_dis	1'b0	Disable PRBS clock
				prbs_clk_en	1'b1	Enable PRBS clock
	[7]	sqwgen_clken		sqwgen_clk_dis	1'b0	Disable square wave clock
				sqwgen_clk_en	1'b1	Enable square wave clock
0x7	[7:5]	prbs_gen_pat	0x8	prbs_15	3'b010	Enable PRBS15 in 64-bit mode
				prbs_23	3'b100	Enable PRBS23 in 64-bit mode
				prbs_31	3'b000	Enable PRBS31 in 64-bit mode
				prbs_9	3'b001	Enable PRBS9 in 64-bit mode
				prbs_dis	3'b000	Disable PRBS in 64-bit mode
0x8	[3:0]	sq_wave_num		sq_wave_1	4'b0001	Enable square wave. One "1"followed by one "0"
				sq_wave_4	4'b0100	Enable square wave. Four "1"followed by four "0"
				sq_wave_8	4'b1000	Enable square wave. Eight "1"followed by eight "0"
				sq_wave_default	4'b0100	
	[4]	prbs_gen_pat	0x7	prbs_15	1'b0	Enable PRBS15 in 64-bit mode
				prbs_23	1'b0	Enable PRBS23 in 64-bit mode
				prbs_31	1'b1	Enable PRBS31 in 64-bit mode
				prbs_9	1'b0	Enable PRBS9 in 64-bit mode
				prbs_dis	1'b0	Disable PRBS in 64-bit mode
	[6:5]	tx_pma_data_sel	0x6	prbs_pat	2'b00	Enable PRBS generator
				sq_wave_pat	2'b00	Enable square wave generator
0x110	[2:0]	ser_mode		sixty_four_bit	3'b011	64-bit mode
				ten_bit	3'b100	10-bit mode

Enabling the PRBS Data Checker

You must perform a sequence of read-modify-writes to the Transceiver Native PHY reconfiguration interface to enable the PRBS checker. You must perform read-modify-writes to offsets: 0xA, 0xB, 0xC, and 0x13F. To enable the PRBS checker, follow these steps:

1. Perform a read-modify-write to offset 0xA with a value of 1'b1 to bit[7].
2. Perform a read-modify-write to offset 0xB according to the PRBS Generator and Square Wave Offsets table. For PRBS9, PRBS15, or PRBS23, write a one to either bit[7], bit[6], or bit[5], respectively. Write to bit[3:2] with the counter threshold before the `rx_prbs_done` signal goes high.
3. Perform a read-modify-write to offset 0xC with a value of 1'b1 to bit[0] for PRBS31, otherwise 1'b0 of other PRBS patterns. Write the other bits according to **Table 6-14**.
4. Perform a read-modify-write to offset 0x13F according to **Table 6-14**.
5. Perform a reset.
   To disable the PRBS verifier write the original values back into the read-modify-write offsets listed above.

Table 6-14: PRBS Checker Offsets

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Other Offsets	Attribute Encoding	Bit Encoding	Description
0xA	[7]	prbs_clken		prbs_clk_dis	1'b0	Disable PRBS checker
				prbs_clk_en	1'b1	Enable PRBS checker

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Other Offsets	Attribute Encoding	Bit Encoding	Description
0xb	[3:2]	rx_prbs_mask		prbsmask1024	2'b11	1023
				prbsmask128	2'b00	Counter threshold to 127
				prbsmask256	2'b01	255
				prbsmask512	2'b10	511
	[7:5]	prbs_ver	0xC	prbs_15	3'b010	Enable PRBS15 in 64-bit mode
				prbs_23	3'b100	Enable PRBS23 in 64-bit mode
				prbs_31	3'b000	Enable PRBS31 in 64-bit mode
				prbs_9	3'b001	Enable PRBS9 in 64-bit mode
				prbs_off	3'b000	Disable PRBS in 64-bit mode

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Other Offsets	Attribute Encoding	Bit Encoding	Description
0xC	[0]	prbs_ver	0xB	prbs_15	1'b0	Enable PRBS15 in 64-bit mode
				prbs_23	1'b0	Enable PRBS23 in 64-bit mode
				prbs_31	1'b1	Enable PRBS31 in 64-bit mode
				prbs_9	1'b0	Enable PRBS9 in 64-bit mode
				prbs_off	1'b0	Disable PRBS in 64-bit mode
	[1]			force_sig_ok	1'b1	
				unforce_sig_ok	1'b0	
	[2]	rx_signalok_signaldet_sel		sel_sig_ok	1'b1	
	[3]	prbs9_dwidth		prbs9_10b	1'b1	PRBS9 10-bit
				prbs9_64b	1'b0	PRBS9 64-bit
	[3:0]	deser_factor		10	4'b0001	10-bit mode
0x13F				64	4'b1110	64-bit mode

Enabling Pseudo Random Pattern Test Mode

Pseudo Random Pattern is a test mode of the scrambler. You can select two seeds of the Pseudo Random Pattern. The seed produces the pattern in the scrambler, and the `r_tx_data_pat_sel` is the data pattern the scrambler scrambles. There are two choices: all 0's or two local fault ordered sets. The Pseudo Random Pattern shares the error signal with PRBS. There is also an error count available. Pseudo Random Pattern is only available when the scrambler is enabled. You must perform a sequence of read-modify-writes to the

reconfiguration interface to enable the Pseudo Random Pattern. The read-modify-writes are required to offsets: 0x82, 0x97, and 0xAC. To enable the Pseudo Random Pattern, complete the following steps:

1. Perform a read-modify-write to offset 0x82X.
2. Perform a read-modify-write to offset 0x97.
3. Perform a read-modify-write to offset 0xAC.
4. Perform a reset.
   To disable the PRBS verifier write the original values back into the read-modify-write offsets listed above.

Table 6-15: Pseudo Random Pattern Test Mode Offsets

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Bit Encoding	Description
0x72	[7:0]	r_tx_seed_a[7:0]		Seed A value bit[7:0]
0x73	[7:0]	r_tx_seed_a[15:8]		Seed A value bit[15:8]
0x74	[7:0]	r_tx_seed_a[23:16]		Seed A value bit[23:16]
0x75	[7:0]	r_tx_seed_a[31:24]		Seed A value bit[31:24]
0x76	[7:0]	r_tx_seed_a[39:32]		Seed A value bit[39:32]
0x77	[7:0]	r_tx_seed_a[47:40]		Seed A value bit[47:40]
0x78	[7:0]	r_tx_seed_a[55:48]		Seed A value bit[55:48]
0x79	[1:0]	r_tx_seed_a[57:56]		Seed A value bit[57:56]
0x7A	[7:0]	r_tx_seed_b[7:0]		Seed B value bit[7:0]
0x7B	[7:0]	r_tx_seed_b[15:8]		Seed B value bit[15:8]
0x7C	[7:0]	r_tx_seed_b[23:16]		Seed B value bit[23:16]
0x7D	[7:0]	r_tx_seed_b[31:24]		Seed B value bit[31:24]
0x7E	[7:0]	r_tx_seed_b[39:32]		Seed B value bit[39:32]
0x7F	[7:0]	r_tx_seed_b[47:40]		Seed B value bit[47:40]
0x80	[7:0]	r_tx_seed_b[55:48]		Seed B value bit[55:48]
0x81	[1:0]	r_tx_seed_b[57:56]		Seed B value bit[57:56]
0x82	[0]	r_tx_data_pat_sel	1'b0	2 local faults
			1'b1	0's
	[1]	r_tx_test_pat_sel	1'b0	Pseudo Random
			1'b1	Square Wave
	[3]	r_tx_test_en	1'b1	
0x97	[2]	r_rx_test_en	1'b1	
0xAC	[0]	r_rx_test_pat_sel	1'b0	Pseudo random
			1'b1	Square wave

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Bit Encoding	Description
0xD7	[7:0]	random_err_cnt[7:0]		Error count
0xD8	[7:0]	random_err_cnt[7:0]		

Enabling PRBS Pattern Inversion

You can enable pattern inversion for PRBS data leaving or entering the PRBS data pattern generator and checker, respectively. The following table shows the offsets to invert the either the generator or checker. To invert the PRBS pattern leaving the PRBS generator or checker, follow these steps:

- To invert the PRBS pattern leaving the PRBS generator, perform a read-modify-write to bit[2] with a value of 1'b1 to offset 0x7.
- To invert the PRBS pattern leaving the PRBS checker, perform a read-modify-write to bit[4] with a value of 1'b1 to offset 0xA.

Table 6-16: PRBS Pattern Inversion Offsets

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Bit Encoding	Description
0x7	[2]	tx_static_ polarity_ inversion	1'b0	Disables static polarity inversion
			1'b1	Enables static polarity inversion
0xA	[4]	rx_static_ polarity_ inversion	1'b0	Disables static polarity inversion
			1'b1	Enables static polarity inversion

Note: Disable the inversion bit on the TX and RX to prevent normal data traffic from being inverted while entering or leaving the PCS.

Unsupported Features

The following features are not supported by either the Transceiver Native PHY IP or the PLL IP reconfiguration interface:

- Reconfiguration from bonded configuration to a non-bonded configuration, or vice versa
- Reconfiguration from bonded protocol to another bonded protocol
- Reconfiguration from PCIe (with Hard IP) to PCIe (without Hard IP) or non-PCIE bonded protocol switching
- Switching between bonding schemes, such as xN to FB compensation
- Connecting the Transceiver Native PHY IP to any other transceiver IP
- Master CGB reconfiguration
- Switching between two master CGBs
- Data rate switching by reconfiguring the CGB local divider
- Serialization factor changes on bonded channels
- TX PLL switching on bonded channels

Note: Transceiver Native PHY IP non-bonded configuration to another Transceiver Native PHY IP non-bonded configuration is supported.

Transceiver and PLL Address Map

The transceiver and PLL address map provides a list of available PCS, PMA, and PLL addresses that you can use to change or reconfigure the functional behavior of the transceiver or PLL, respectively. Use the transceiver and PLL address map with the Avalon interface. The address map is provided as an Excel spreadsheet for easy search and filtering.

Related Information

Transceiver and PLL Address Map

Document Revision History 7

2013.12.02

UG-A10XCVR Subscribe Send Feedback

The table below lists the revision history for this user guide.

Data	Version	Changes
December 2013	2013.12.02	Initial release.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered